GMS Single-Page Requirements

This page offers a single-page rendering of the requirements.

1 Introduction

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Requirements	Links to Domains
1.1	Definitions (/gms/policies/domains/introduction#definitions)

This document enumerates the *technical requirements* (“GMS Requirements”) that *Android devices shipping with the Google Mobile Services* ("DEVICE") must meet, per commercial agreement between Google and the *partner company* (“Partner”).

The use of “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” is per the IETF standard defined in RFC2119 (http://www.ietf.org/rfc/rfc2119.txt).

Per commercially agreed terms, a DEVICE MUST comply with the Android Compatibility Definition Document (CDD) (http://source.android.com/compatibility/cdd.html) and pass applicable testing suites, such as CTS (http://source.android.com/compatibility/cts/index.html), CTS Verifier (http://source.android.com/compatibility/cts/verifier.html), and GTS (/gms/testing/gts).

In this document, *GMS* refers to the Google software that is designed to be shipped on Android-compatible handheld and tablet DEVICE implementations as defined in the CDD. Approval from Google is REQUIRED if the Partner intends to ship GMS on different DEVICE form factors.

If Google and the Partner entered into any commercial agreement that defines additional requirements for a DEVICE, both the GMS Requirements and the requirements in such an agreement MUST be met. However, if there are conflicts, the Partner MUST seek a resolution with Google.

GMS Requirements are grouped by the following categories:

- **Ecosystem freshness:** Requirements to ensure that DEVICEs are shipped/updated with the latest Android OS versions, and security patches.
- **Ecosystem freshness - Treble:** How a DEVICE can comply with Treble, a modular Android architecture that allows easy upgrades to higher releases of the Android OS and Mainline (aka Updatability), which aims to make core parts of Android easily updatable independent from full system OTAs.
- **Ecosystem freshness - Mainline updates (GPsu):** How a DEVICE can comply with Mainline, which aims to make core parts of Android easily updatable independent from full system OTAs.
- **Regulatory requirements:** Requirements to meet different regulatory requirements (Russia, EEA, Turkey, Latchsky, etc.)
- **Best Google experience:** Requirements for how Google-branded apps and services should be placed/integrated/presented in general on DEVICEs.
- **Best Google experience - app specific:** Requirements for a particular Google-branded app or service.
- **Best user experience:** Requirements for providing the best Android experience to users.

- **Protect Users:** Requirements on how a DEVICE must protect the user's security and privacy.
- **Android Go:** The requirements that are only applicable for a category of low-cost DEVICEs.
- **Android Enterprise:** How a DEVICE should behave when it's set up for enterprise use cases.
- **Platform policies:** Platform policies such as Location Privacy Policy, System Behavior, and Emergency Location Bypass Policy.
- **MBA policies:** Policies for mobile bundled apps (MBAs) such as Content, Pre-grants Permission, and SMS/Call Log.

1.1 Definitions

- **DEVICE:** A group of Android device models sharing similar hardware characteristics, identified by a distinct combination of `android.os.Build.BRAND` and `android.os.Build.DEVICE` values in their software BUILDs.
- **PRODUCT:** A physical Android device model that's identified by a distinct `android.os.Build.PRODUCT` value and the DEVICE it belongs to. A DEVICE can include multiple PRODUCTs, for example, a version for a regional market, or for a specific carrier.
- **mobile bundled apps (MBAs):** Apps that a device manufacturer or mobile network operator bundles by preloading in the shipping software flashed onto the device, or apps that are distributed over the air and installed on a user's device at any time (and not just during or shortly after the setup phase) by the device manufacturer’s services or other preloaded third-party installers, without the user explicitly choosing to download and install them. Even when the user is presented with an option to install an app, the app is considered an MBA if the user’s choice isn't explicit. The following are examples of when apps are considered MBAs:
  - The user is presented with a list of apps to be installed but with checkboxes selected by default.
  - The user is presented with a notification to install an app but with a single-click to install, without an alternative option presented equally.

  Any MBA that is also updated through Google Play (that is, an app on Google Play with a matching package name and the same signature) MUST adhere to the Google

Play developer policy (https://play.google.com/about/developer-content-policy/#!?modal_active=none).

- **default launcher:** A user interface that is (1) initiated by the `KEYCODE_HOME` (https://developer.android.com/reference/android/view/KeyEvent#KEYCODE_HOME) key event in Android (for example, pressing the home button or any applicable gesture intended to invoke the home function as defined by the CDD), or (2) started after initial bootup or subsequent power up of the device.
- **default home screen:** The layouts of icons and widgets (prior to any changes made by the user) that are made visible by the OOB default launcher app across all displays on a device immediately after bootup and unlocking from the lock screen. If a device has multiple displays, then that device has (i) multiple default home screens if it has multiple layouts, such as a foldable device; and (ii) only one default home screen if it has only a single layout across multiple displays.
- **software BUILD:** The software configuration for a PRODUCT identified by a unique BUILD fingerprint.
- **new software BUILD:** A new software BUILD for an existing approved PRODUCT. Such BUILDs are typically created to update the software of existing DEVICE units.
- **Work Profile:** A Work Profile (also known as a *managed profile*) is a secondary Android user with additional properties around device management. A Work Profile creates a segregated and secure space for managed data (such as corporate data). The administrator of the Work Profile has full control over scope, ingress, and egress of data as well as its lifetime.
- **Personal Profile:** In order to support deployment models where work and personal apps live together on the same device, Android puts them into different containers, or *profiles*. On a device with a Work Profile, the Personal Profile refers to the primary Android user on the device. A Personal Profile is an Android user profile associated with a personal user account which securely isolates apps and data from the Work Profile.
- **Personal Profile data:** For the purposes of the MBA Personal Profile data privacy policy section (/gms/policies/domains/mba#mba-personal-profile-privacy), Personal Profile data on DEVICEs is any information collected or processed from the Personal Profile or about individuals who interact with the primary Android user (Personal Profile). Personal Profile data might include information that is collected from users, created by users, received on behalf of users, generated about users, as well as information observed or derived from Personal Profile data. Examples of Personal Profile data include:

    - Package names of applications in the Personal Profile
    - Statistics regarding app usage in the Personal Profile
    - Files or app data saved in the Personal Profile
    - Personalization data such as dictionaries in the Personal Profile
    - Details of telephony initiated from or received in the Personal Profile
- **Low RAM device:** A device implementation that returns `true` for the public method `ActivityManager.isLowRamDevice()`.
- **Protection levels:** Protection levels referred here are those listed in `protectionLevel` (https://developer.android.com/reference/android/R.attr#protectionLevel) constants.
- **system or special UIDs:** All user IDs defined in AOSP `Process.java` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/java/android/os/Process.java)

  .
- **system UID:** Android UID `android.uid.system (1000)` as defined in AOSP `Process.java` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/java/android/os/Process.java)

  which is used by the system server to run.
- **custom system permissions:** Custom system permissions are those system permissions that are defined by the OEMs in their system image and not defined in AOSP `AndroidManifest.xml` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml)

  .
- **custom Android APIs:** Android Platform APIs that are defined by the OEMs in their system image and aren't defined in AOSP APIs (https://developer.android.com/reference).
- **custom system UIDs:** System UIDs that are defined by the OEMs in their system image and not defined in AOSP `Process.java` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/java/android/os/Process.java)

  .
- **UI automation:** `UiAutomation` (https://developer.android.com/reference/android/app/UiAutomation) is an accessibility framework that enables android applications to provide and consume programmatic

information about user interfaces (UIs). It provides programmatic access to most UI elements on the screen.

- **forced or keep enabled apps:** Apps that users have no means to disable (including through the default mechanism provided by the AOSP Settings app or any alternative provided).
- **stub APK:** A bare bones APK that must be signed and included in the device's factory ROM. It contains `AndroidManifest.xml` and MUST be updated before its first use.
- **headless apps:** Apps with no icon in the launcher.
- **persistent notice:** A notice that must be dismissed by the user (such as a user only dismissible notification, system dialog, app popup) and not closed programmatically.
- **permission sharing apps:** Apps sharing resources and capabilities with other apps using Custom app permissions (https://developer.android.com/guide/topics/permissions/defining), shared UIDs (https://developer.android.com/guide/topics/manifest/manifest-element#uid), Android `Service` (https://developer.android.com/reference/android/app/Service) (enter app service), same signatures/certificates, or any framework modifications or mechanisms.
- **capability:** The right to perform an operation. Note this includes returning data, but also sensitive operations that don't return any data, for example, the right to send a premium SMS or write to contacts.
- **personal or sensitive capability:** The capabilities of an app that give access to personal and sensitive (/gms/policies/domains/mba#personal-sensitive-info) capabilities as defined in the MBA User data policy (/gms/policies/domains/mba#mba-user-data) and grant privileges (for example, installing apps, sending premium SMS etc.).

Start new requirements for 12

- **VENDOR:** A chipset manufacturer, or organization supplying the hardware chips and the software, such as a board support package (BSP), that is compliant to run Android.
- **CHIPSET:** A group of chips integrated to work together, identified by a distinct CHIPSET IDENTIFIER and developed by a VENDOR.
- **CHIPSET IDENTIFIER:** The VENDOR specific model name or chipset family identifier that resides within a DEVICE. The identifier may be a compound name and may be versioned. A DEVICE must have exactly one CHIPSET IDENTIFIER.

End new requirements

1.2 Android branding

"Powered by Android" bootup screen

The purpose of the "Powered by Android" bootup co-branding is to make it clear to consumers globally that the device is running a compatible version of Android OS.

[GMS-1.2-001] The "Powered by Android" bootup co-branding MUST be displayed during the bootup sequence according to the guidelines published on the Partner Marketing Hub (https://partnermarketinghub.withgoogle.com/) site.

- **[GMS-1.2-001.001]** DEVICEs launching with Android 10 or higher, and DEVICEs launching on or after January 1, 2020 with any Android version MUST use the bootup co-branding.
- **[GMS-1.2-001.002]** DEVICEs upgrading to Android 10 and DEVICEs applying MRs to Android 10 or lower after September 3, 2019 are STRONGLY RECOMMENDED to use the bootup co-branding.

2 Ecosystem Freshness

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Requirements	Links to Domains
2.1	Launch approval windows (#launch-approval-windows)
2.4	Resume on reboot (#resume-on-reboot)

2.1 Launch approval windows

To ensure that GMS functions consistently and properly for users in the future, Google approves GMS distribution only on new Android PRODUCTs shipping on higher platform releases.

Google approves GMS distribution only for:

- A new Android make and model, defined by a distinct `android.os.Build.DEVICE`, running the latest major Android OS release.
- A software BUILD running one of the two latest major releases of the Android OS.

For more information about distribution BUILDs and approval, see Overview and Launch Approval (/gms/testing/overview).

GMS approval windows by release

Select the AOSP version range for launch approval dates.

- [x] 11 through 12L
- [ ] 8.1 through 10
- [ ] 7.0 through 8.0

OS version (API level)	AOSP release	Approval window closed for	GMS approval expiration dates[1]	Security patch support dates

		New DEVICE	New PRODUCT	New S/W BUILD[4]	Handheld devices	Handheld Go devices[3] and tablet devices	Date of termination of security backport support	ACK support end date[2]
12L (32)	Mar 7, 2022	Jan 31, 2023	N/A	Android U launch	Dec 31, 2023	Dec 31, 2024	Mar 31, 2025	4.19.191+: Dec 2024 5.4.86+: Dec 2025 5.10.43+: Dec 2026
12 (31)	Oct 4, 2021	Jan 31, 2023	N/A	Android U launch	Dec 31, 2023	Dec 31, 2024	Mar 31, 2025	4.19.191+: Dec 2024 5.4.86+: Dec 2025 5.10.43+: Dec 2026
11 (30)	Sep 8, 2020	~~Mar 31, 2022~~ Apr 30, 2022	N/A	Android T launch	Mar 31, 2023	Mar 31, 2024	Feb 29, 2024	4.14.180+: Jan 2024 4.19.110+: Dec 2024 5.4.61+: Dec 2025

1 See the GMS approval expiration dates (/gms/policies/overview/approval-expiration-dates) for additional details.

2 The support window end date is the latter of the upstream kernel support policy or the Android common kernel support policy. These dates will be updated as needed if there are changes to the upstream kernel timelines.

3 A *Go device* is a class of devices defined with a limited size of RAM (low-memory device) and required to return `true` for the public API `ActivityManager.isLowRamDevice()`. See Android Go (/gms/policies/domains/go) for more details.

4 The software configuration for a PRODUCT is identified by a unique BUILD fingerprint.

To assist Partners in meeting approval window deadline, we provide early access to upcoming platform software through the Platform Development Kit (PDK) and other

software distributions, as well as additional resources (guideline documents, CDD summary, etc.) through the Early Access Program site (https://partner.android.com/eap).

AOSP receives updates for platform security issues documented in the public Android Security Bulletins for 3.5 years from when the OS version is released publicly. Security updates pass through a basic level of testing and then are published to AOSP on the same schedule as the public Android Security Bulletin during the OS support period.

2.2 Android platform fixes

[GMS-2.2-001] Software BUILDs running Android 8.0 or higher that aren't a Security Maintenance Release (SMR) or Emergency Maintenance Release (EMR) MUST apply all mandatory fixes from advisories listed in the GMS Help Center Bulletins (/gms/policies/bulletins), up to 60 days prior to BUILD approval.

2.3 Security Patch

[GMS-2.3-001] Software BUILDs, including those with Launch Release (LR) and Maintenance Release (MR) BUILDs for PRODUCTs launched within the last 24 months, MUST have all security patches applied that were issued publicly (https://source.android.com/security/bulletin/) more than 60 days ago from the date it's submitted to request Google's approval. This means that all required security patches in a monthly partner security bulletin (https://groups.google.com/forum/#!forum/android-partner-security) have to be applied within 90 days from its issue.

2.4 Resume on reboot

Start new requirements for 12

[GMS-2.4-001.001] DEVICEs running Android 12 MUST implement the resume on reboot service: `android.service.resumeonreboot.ResumeOnRebootService`.

[GMS-2.4-001.002] The SIM PIN status MUST be restored after reboots initiated by the `ResumeOnRebootService1`.

End new requirements

~~**[GMS-2.4-001.001]** For DEVICEs launching with Android 11, it's STRONGLY RECOMMENDED to implement reboot escrow HAL (`android.hardware.rebootescrow.IRebootEscrow`) and use the new Resume on reboot APIs in `RecoverySystem` for OTA updates:~~

~~`android.os.RecoverySystem#prepareForUnattendedUpdate()`~~
~~`android.os.RecoverySystem#rebootAndApply()`~~
~~`android.os.RecoverySystem#clearPrepareForUnattendedUpdate()`~~

~~**[GMS-2.4-001.002]** DEVICEs upgrading to Android 11 SHOULD support this requirement if the underlying SoC can support this HAL.~~

3 Treble

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Vendor support requirements compliance

DEVICEs MUST comply with the requirements as described below. The relevant requirements are determined by VENDOR names, CHIPSET IDENTIFIERs, and Android OS versions.

[GMS-3-001.001] and **[GMS-3-001.002]** DEVICEs MUST follow the requirements that are described in a Vendor Software Requirement (VSR) document.

The VSR document to refer to is determined by the CHIPSET that resides within a DEVICE as shown in the table below.

[GMS-3-001.003] When a chipset that is not listed in the GRF table (https://drive.google.com/file/d/10MJdhQZJeEGbTjxLAW8liMrn39-ksb3y/view?usp=sharing&resourcekey=0-kVfHNRRZXm8_-x0u1kjyCw)

is used, then:

- DEVICEs launching with Android 12 MUST meet requirements in VSR-Android 12.
- DEVICEs upgrading to Android 12 MAY optionally meet requirements in VSR - Android 12

CHIPSET	Requirement
A CHIPSET that is in Vendor Freeze Program and frozen on Android 11	Vendor Software Requirements for Android 11 (/gms/policies/vsr/vsr-11)
All other CHIPSETs	Vendor Software Requirements for Android 12 (/gms/policies/vsr/vsr-12)

4 Mainline Updates

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Update in upcoming GMS requirements release

Release	Requirements	Links to requirements
Android 11	4.1	Platform requirements (updated Nov 15, 2021) (#platform-requirements)
March 2022	4.1.2	Soft restart (#soft-restart)

March 2022	4.2.2	Module preload requirements for Low RAM devices (#module-preload-requirements-for-low-ram-devices)
March 2022	4.5	Mainline UX requirements (#mainline-ux-requirements)

Project “Mainline updates” delivers updates to system components through the Google Play Store. Mainline modules are encapsulated in an APK or APEX container. The requirements in this section are applicable for all DEVICEs running Android 10 or higher, except when stated otherwise. Project Mainline updates is referred to as Google Play system update (GPSU).

4.1 Platform requirements

[GMS-4.1-001] New DEVICEs launching with Android 10 or higher, including Low RAM devices, MUST implement the following platform features to support Mainline modules and module updates through the Google Play Store:

- APEX (https://source.android.com/devices/tech/ota/apex)
- User Data Checkpointing (https://source.android.com/devices/tech/ota/user-data-checkpoint)
- Stable AIDL (https://source.android.com/devices/bootloader/stable-aidl)
- File-based Encryption (https://source.android.com/security/encryption/file-based)

[GMS-4.1-002] DEVICEs running Android 11 or higher MUST NOT have the `sys.init.updatable_crashing` property set as `1` at the end of a normal boot sequence.

Note: When multiple native service crashes are detected in a short period of time, the property is set to `1`, which triggers rollback of the active Mainline train. Partners are expected to suppress any unexpected native crashes so that this property can serve as a clear signal for rollback.

[GMS-4.1-003] DEVICEs running Android 12 or higher MUST implement the following platform features to support Mainline modules and module updates through the Google Play Store:

- Server-based resume on reboot (/gms/building/mainline/eap/ror)

- Reboot readiness detection
- Compressed APEX file preloads (https://source.android.com/devices/tech/ota/apex#compressed-apex)
- De-duplication of shared libraries

[GMS-4.1-004] DEVICEs running Android 11 MUST apply all the patches mentioned in the Partner Security Advisory—2021-10 (Mainline APEX Reset), and declare `com.google.android.mainline.patchlevel.1` as one of its system features.

Note: For Android 11 devices, Google Play delivers future Mainline trains to devices with the patches and the system features only.

End new requirements

4.1.1 Multiple trains support

[GMS-4.1.1-001] DEVICEs running Android 11 or higher (including upgrades) MUST support multiple trains (/gms/building/mainline/releasing/multiple-trains).

[GMS-4.1.1-002] Partners MUST NOT use the multiple trains support to update their own code. This feature MUST only be used by the Google Play Store client.

4.1.2 Soft restart

Soft restart (https://source.android.com/devices/bootloader/soft-restart) (also known as user space reboot) is introduced in Android 11 to restart the userspace components and install staged APEX or APK files, without prompting the user, while keeping the Credential Encrypted (CE) storage unlocked.

Start new requirements for 12

~~**[GMS-4.1.2-001]** DEVICEs launching with Android 11 or higher that support updatable APEX are STRONGLY RECOMMENDED to support soft restart. DEVICEs upgrading to Android 11 or higher MAY optionally support this feature.~~

[GMS-4.1.2-001] DEVICEs launching with Android 11 that support updatable APEX are STRONGLY RECOMMENDED to support soft restart. DEVICEs upgrading to Android 11 MAY optionally support this feature.

End new requirements

[GMS-4.1.2-002] DEVICEs that support soft restart MUST be compliant with requirements in Soft Restart documentation (https://source.android.com/devices/bootloader/soft-restart).

4.1.3 Rollback of CE and DE data

[GMS-4.1.3-001] DEVICEs running Android 10 or higher MUST include the CE and DE user data of an APK as part of the data being saved and restored during rollback operation performed by `RollbackManager`.

Note: `RollbackManager` in AOSP Android 12 contains the reference implementation of handling CE and DE data during rollback, but wasn't part of `RollbackManager` of Android 10 or Android 11. Refer to A-169594054 (/gms/policies/bulletins/nov-2020#A-169594054) to meet this requirement for Android 10 and 11 devices.

4.2 Preload requirements

[GMS-4.2-001] DEVICEs running Android 10 or higher MUST preload Google-signed modules in compliance with the rest of this section and its subsections, and Module-specific requirements (#module-specific-requirements).

[GMS-4.2-002] DEVICEs that launched with Android 9 or lower and implemented full disk encryption (FDE) MUST NOT preload any Google-signed updatable APEX modules when they upgrade to Android 10 or higher.

[GMS-4.2-003] DEVICEs that preload any Google-signed updatable APEX modules MUST set `ro.apex.updateable` to true and MUST NOT set `TARGET_FLATTEN_APEX` to `true`.

Note: See the Mainline Partner Integration Guide (/gms/building/mainline/integrating) for more details.

[GMS-4.2-004] When preloading Google-signed updatable APEX modules, DEVICEs MUST preload one of the Google-signed module trains that have been approved by Google for preloading. See Module Releases (/gms/building/mainline/releasing/module-releases) for the latest approved module trains. However, an Initial Release (IR) build for a new PRODUCT SHOULD preload the latest approved module trains.

[GMS-4.2-005] Subsequent Security Maintenance Release (SMR) builds and Maintenance Release (MR) builds are NOT REQUIRED to preload updated versions of Google-signed modules.

- Permission Controller - APK
- Ext Services - APK

The GMS-4.2-005 requirement doesn't apply to the following modules when they're referenced in the Android Security Bulletin (https://source.android.com/security/bulletin).

Note: The Android Security Bulletin can require a newer version of the above modules to meet a specific Security Patch Level (SPL). Refer to Module-specific requirements (#module-specific-requirements) for more details.

[GMS-4.2-006] DEVICEs preloading any STRONGLY RECOMMENDED or OPTIONAL Google-signed modules MUST NOT switch back to non-Google-signed modules.

4.2.1 Module preload requirements

[GMS-4.2.1-001] DEVICEs that aren't Low RAM devices, MUST preload Google-signed modules with respect to the following table.

Module name	Type	Android 12 or higher[4] (#4)	Android 11 1 (#1)	Android 10 new launches
adbd `com.google.android.adbd`	APEX	MUST[2]	MUST[2]	unsupported

Android Neural Network API `com.google.android.neuralnetworks`	APEX MUST[2]	MUST[2]	unsupported
ART `com.google.android.art`	APEX MUST[2]	unsupported	
Captive Portal Login[3] `com.google.android.captiveportallogin`	APK MUST	MUST	STRONGLY RECOMMENDE
Cell Broadcast[3] `com.google.android.cellbroadcast`	APEX MUST[2]	MUST[2]	unsupported
Conscrypt `com.google.android.conscrypt`	APEX MUST[2]	MUST[2]	STRONGLY RECOMMENDE
Device Scheduling `com.google.android.scheduling`	APEX MUST[2]	unsupported	
DnsResolver `com.google.android.resolv`	APEX MUST[2]	MUST[2]	STRONGLY RECOMMENDE
Documents UI `com.google.android.documentsui`	APK MUST	MUST	MUST
Dynamic Common Libraries `com.google.mainline.primary.libs`	APEX MUST[2]	unsupported	
ExtServices - APK[3] `com.google.android.ext.services`	APK MUST	MUST	MUST
ExtServices - APEX[3] `com.google.android.extservices`	APEX MUST[,2]	MUST[,2]	unsupported
Ipsec `com.google.android.ipsec`	APEX MUST[2]	MUST[2]	unsupported
Media Codecs[3] `com.google.android.media.swcodec`	APEX MUST[2]	MUST[2]	MUST
Media Framework components[3]	APEX MUST[2]	MUST[2]	MUST

com.google.android.media				
Media Provider com.google.android.mediaprovider	APEX	MUST[2]	MUST[2]	unsupported
Module Metadata com.google.android.modulemetadata	APK	MUST	MUST	MUST
NetworkStack[3] com.google.android.networkstack	APK	MUST	MUST	STRONGLY RECOMMENDE
Network Stack Permission Configuration[3] com.google.android.networkstack.permissionconfig	APK	MUST	MUST	STRONGLY RECOMMENDE
Permission Controller - APK[3] com.google.android.permissioncontroller	APK	MUST	MUST	MUST
Permission Controller - APEX[3] com.google.android.permission	APEX	MUST[,2]	MUST[,2]	unsupported
Sdk Extensions com.google.android.sdkext	APEX	MUST[2]	MUST[2]	unsupported
Statsd com.google.android.os.statsd	APEX	MUST[2]	MUST[2]	unsupported
Telemetry TVP com.google.mainline.telemetry	APK	MUST	MUST	unsupported
Tethering[3] com.google.android.tethering	APEX	MUST[2]	MUST[2]	unsupported
Time Zone data[3] com.google.android.tzdata	APEX	MUST NOT	MUST NOT	MUST
Time Zone data 2[3] com.google.android.tzdata2	APEX	MUST NOT	MUST[2]	unsupported
Time Zone data 3[3] com.google.android.tzdata3	APEX	MUST[2]	unsupported	

WiFi[3] `com.google.android.wifi`	APEX	OPTIONAL	OPTIONAL	unsupported

[1] All DEVICEs running Android 11, including launches and upgrades.

[2] Only applicable to DEVICEs supporting updatable APEX.

[3] See Module specific requirements (/gms/policies/domains/mainline#module-specific-req) for more details.

[4] All DEVICEs running Android 12 or higher, including launches and upgrades.

4.2.2 Module preload requirements for Low RAM devices

[GMS-4.2.2-001] Low RAM DEVICEs that run Android 10, 11, 12, or 12L MUST preload the following Google-signed modules only, and MUST NOT preload any other Google-signed modules.

- Module Metadata
- Permission Controller - APK
- Ext Services - APK

4.3 Integrity, testing, and upgrading requirements

Google-signed Mainline modules are released on a regular cadence as a train. A module train is a set of modules that are updated (downloaded and installed) at the same time. For more details, see the summary of releases and schedules (/gms/building/mainline/releasing).

[GMS-4.3-001] DEVICEs MUST NOT remove, modify, or replace the Mainline modules.

[GMS-4.3-002] DEVICEs MUST NOT prevent the update of a module train through Google Play Store.

[GMS-4.3-003] DEVICEs MUST add an overlay configuration for the metadata provider, by setting `config_defaultModuleMetadataProvider` to `com.google.android.modulemetadata`.

[GMS-4.3-004] Partners MAY customize modules through overlays (/gms/building/mainline/rros) or other runtime mechanisms, and such customizations MUST be performed in a manner that is permitted and documented.

[GMS-4.3-005] For any preloaded Google-signed modules, DEVICEs MUST NOT preload any software components that provide functions similar to the preloaded Mainline modules and MUST use the preloaded modules for the intended purpose, unless explicitly permitted by the module-specific requirements below or the module-specific documentation.

4.4 Module-specific requirements

4.4.1 Network framework modules

[GMS-4.4.1-001] On DEVICEs running Android 10 or higher, the following Google-signed network modules MUST be all preloaded all together. Otherwise, on configurations where they aren't mandatory, for example, Low RAM DEVICEs, any of them MUST NOT be preloaded.

- Captive Portal Login
- Network Stack
- Network Stack Permission Configuration

[GMS-4.4.1-002] When preloading Google-signed network modules, partners MUST use the Google-provided `networkstack.x509.pem` file to generate a device system image.

[GMS-4.4.1-003] DEVICEs that don't preload Google-signed network modules (Android 10 or Android 11 Low RAM devices) MUST preload the followings AOSP modules, which MUST be signed with the platform key:

- `InProcessNetworkStack` (in place of `NetworkStack`)
- `PlatformCaptivePortalLogin` (in place of `CaptivePortalLogin`)
- `PlatformNetworkPermissionConfig` (in place of `NetworkPermissionConfig`)
- `com.android.tethering.inprocess` (in place of `Tethering`, for Android 11 or higher)
- `CellBroadcastAppPlatform` (in place of `com.android.cellbroadcast`, for Android 11 or higher)
- `CellBroadcastServiceModulePlatform` (in place of `com.android.cellbroadcast`, for Android 11 or higher)

See the Mainline Partner Integration Guide (/gms/building/mainline/integrating) for more details.

4.4.2 Media codecs and media framework components

For details, see Module: Media (/gms/building/mainline/modules/module-media).

4.4.3 Time Zone Data

[GMS-4.4.3-001] DEVICEs running Android 10 that support updateable APEX MUST preload the Timezone Data module with matching package name `com.google.android.tzdata`.

[GMS-4.4.3-002] DEVICEs running Android 11 that support updateable APEX MUST preload the Timezone Data module with matching package name `com.google.android.tzdata2`.

Note: DEVICEs upgrading from Android 10 to 11 that support updatable APEX MUST switch from `com.google.android.tzdata` to `com.google.android.tzdata2` in the system update image.

[GMS-4.4.3-003] DEVICEs running Android 12 that support updateable APEX MUST preload the Timezone Data module with matching package name `com.google.android.tzdata3`.

Note: DEVICEs upgrading from Android 11 to 12 that support updatable APEX MUST switch from `com.google.android.tzdata2` to `com.google.android.tzdata3` in the system update image.

4.4.4 Permission Controller and Ext Services

In Android 10, Google-signed Permission Controller and Ext Services modules are distributed in APK format and Google Play Store doesn't update them.

Beginning with Android 11, they're distributed in both APK and APEX format. If the APEX versions of these modules are preloaded, Google Play Store updates them along with other Google-signed modules.

[GMS-4.4.4-001] DEVICEs running Android 10, including Low RAM devices MUST preload Google-signed Permission Controller and Ext Services modules in APK format.

[GMS-4.4.4-002] Low RAM DEVICEs running Android 11, 12, and 12L MUST preload Google-signed Permission Controller and Ext Services modules in APK format.

[GMS-4.4.4-003] DEVICEs running Android 11 or higher that don't support updateable APEX MUST preload Google-signed Permission Controller and Ext Services modules in APK

format.

[GMS-4.4.4-004] DEVICEs running Android 11 or higher that support updateable APEX (including upgrades) MUST preload Google-signed Permission Controller and Ext Services modules in APEX format.

Note: DEVICEs upgrading to Android 11 or higher, if they support updatable APEX, MUST switch from Google-signed APK to Google-signed APEX for Permission Controller and Ext Services modules when updating the system image.

4.4.5 Wi-Fi

[GMS-4.4.5-001] On DEVICEs running Android 11 or higher, Wi-Fi is an optional module. However, preloading this module is allowed only for devices that meet the set of technical criteria provided by Google. For more information, contact your technical account manager.

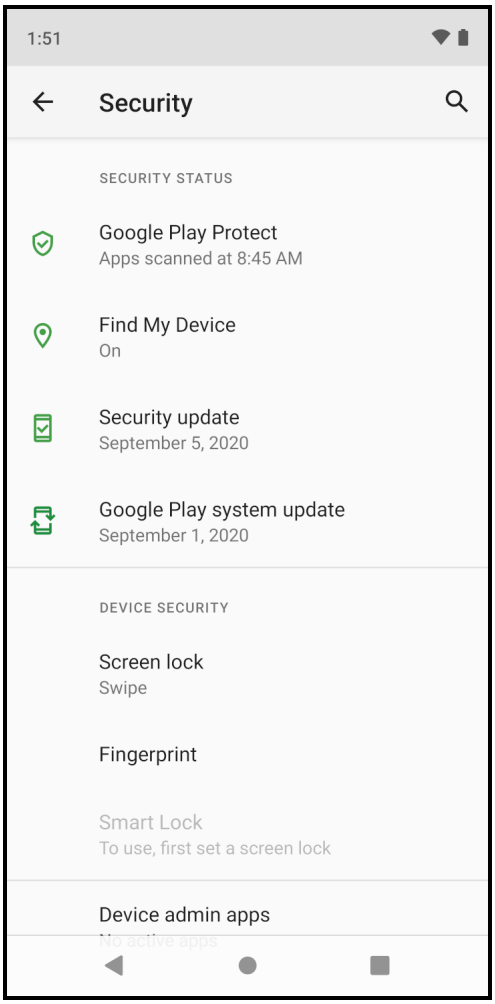
4.4.6 CellBroadcast

[GMS-4.4.6-001] For Tablet DEVICEs running Android 11 or higher that don't provide voice or messaging service, DEVICEs MAY set `config_disable_all_cb_messages` to `true`. When it is turned off, DEVICEs MUST stop forwarding Cellbroadcast messages to CellBroadcast modules and MUST hide WEA-related UI and notifications.

Note: This is for data-only tablet DEVICEs which need to hide WEA-related alerts. CellBroadcast module MUST be preloaded regardless of this configuration.

4.5 Mainline UX requirements

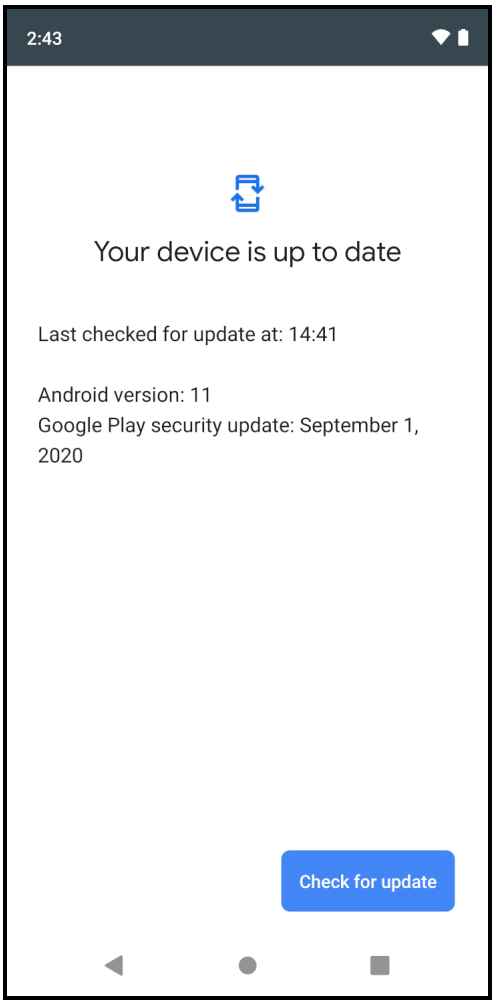
[GMS-4.5-001] DEVICEs MUST show the Mainline train version as a date at the bottom of Security status in addition to the items listed in Security Status in Settings (/gms/building/settings/security/security-status-settings).



Note: DEVICEs MAY add an extra entry point in the Settings menu, in addition to the one in the Security status menu, to allow users to bring up Google Play system update screen as shown in the screenshot for [GMS-4.5-002] (#GMS-4.5-002). Such an entry point MUST be labeled as **Google Play system update**.

Note: On DEVICEs supporting multiple trains, the version string under the menu in the screenshot above shows the oldest version across all the trains.

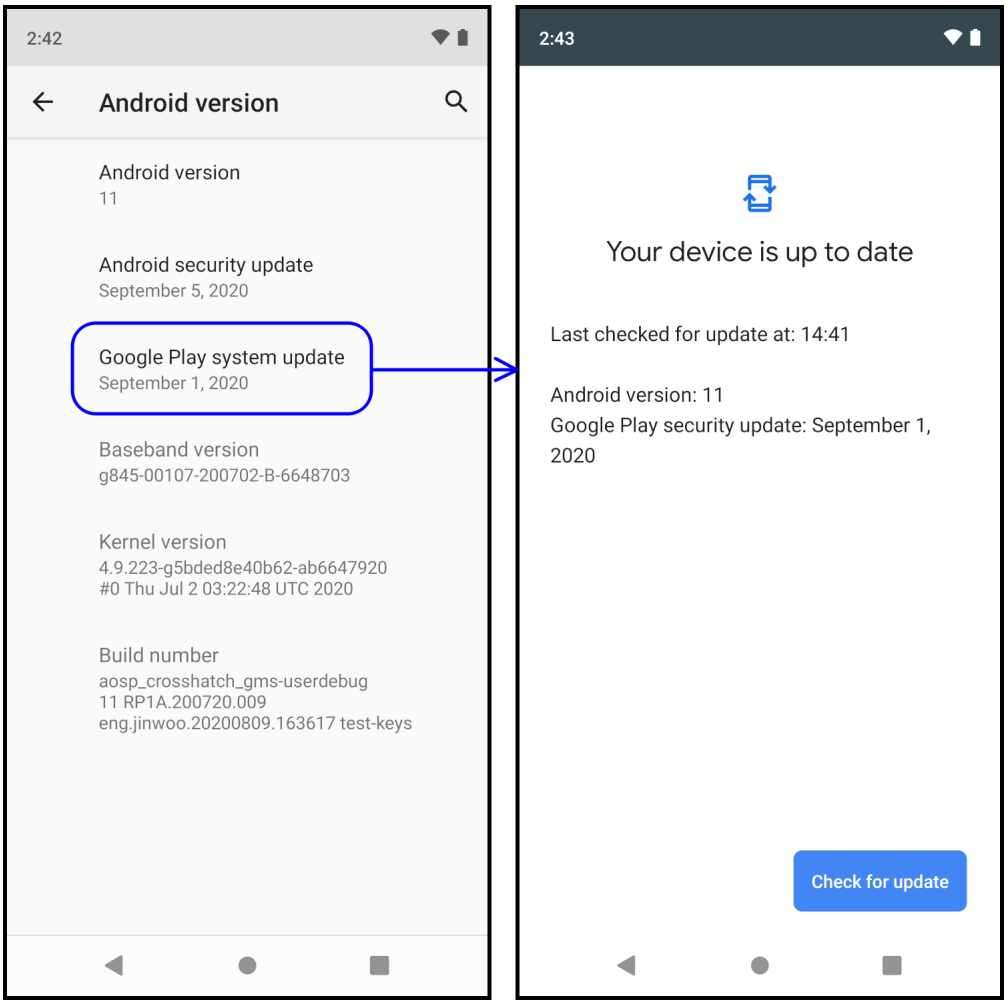
[GMS-4.5-002] Tapping **Google Play system update** MUST show the following activity in the Google Play Store app that lets users manually update the Mainline modules.



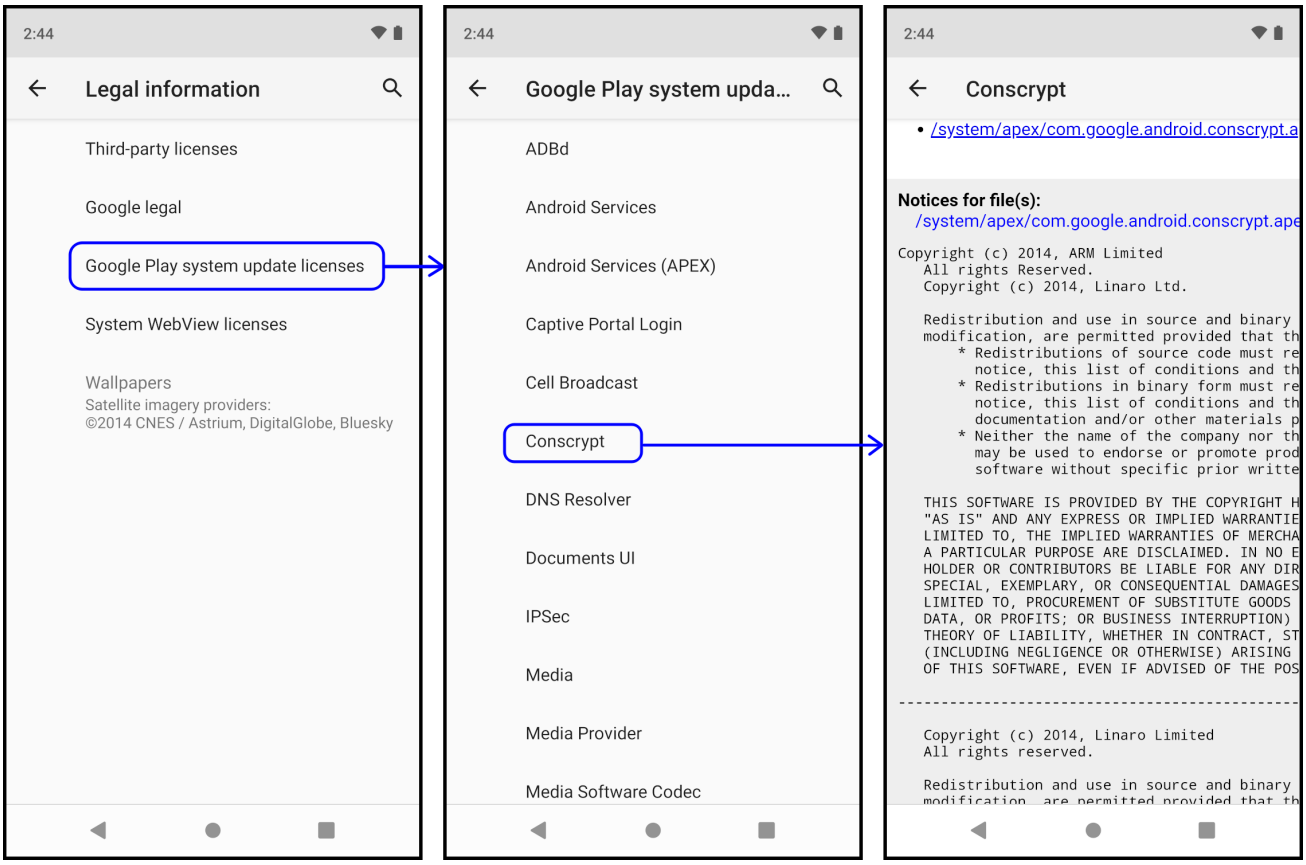
[GMS-4.5-003] DEVICEs supporting Multiple Trains are STRONGLY RECOMMENDED to have a menu named **Google Play system update** under **Settings** > **About Phone** > **Android version** or under a similar preference list. When tapped, it MUST show the train version information screen using an action intent as follows:

```
Intent intent = new Intent("android.settings.MODULE_UPDATE_VERSIONS");
```

The screenshot shows version information per each train supported.



[GMS-4.5-004] Android 10 added `ModuleLicensesActivity` in AOSP Settings app for showing legal information extracted from Mainline modules. This is shown with the title “Google Play system update licenses” as part of legal information activity that resolves `android.settings.TERMS`. **DEVICEs MUST preserve this mechanism.**



Start new requirements for 12

[GMS-4.5-005] For DEVICEs that support soft restart (aka user space reboot) or unattended reboot, while performing the Soft Restart or unattended reboot (identified by "unattended,mainline_update" as reboot_reason) all boot screens and animations:

- MUST have black background with white text or white image only to minimize user disruption.
- MUST NOT make any sound or vibration while performing the Soft Restart.

Note: Refer to the Soft Restarts technical document (https://source.android.com/devices/bootloader/soft-restart) and Soft Restarts UX document (/gms/building/mainline/releasing/soft-restarts) for details on the reference implementation in AOSP and options to customize the animation.

Note. When the reboot is triggered by end user in the screen shown in GMS-4.5-002, reboot_reason will be “userrequested,mainline_update” and this requirement does not apply.

End new requirements

4.6 DeviceConfig API

Android 10 introduces DeviceConfig API (android.provider.DeviceConfig), an important framework component for Project Mainline. The API is used to perform safe roll-outs of new Mainline modules using server-controlled flags.

[GMS-4.6-001] DEVICEs running Android 10 and higher MUST:

- Keep the behavior of the API consistent with that of AOSP
- Grant Google Play services app access to the API by setting its package name as the value for the `config_deviceConfiguratorPackageName` framework resource

[GMS-4.6-002] DEVICEs MUST NOT override any configurations managed by `DeviceConfig` API in any BUILDs for approval.

5 Regulatory Requirements

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Release	Requirements	Links to requirements
March 2022	EEA builds	EEA builds (#eea-builds)

EEA builds

GMS smartphones and tablets sold or distributed in the European Economic Area (EEA) must meet the GMS Requirements for EEA Builds (/gms/policies/restricted/eea-builds).

Start new requirements for 12

EEA builds for regular Android devices running Android 11 or lower, if they didn't preload the Google app, MUST preload the following headless apps:

- GoogleSpeechServices (`com.google.android.apps.speechservices`)
- GoogleActionsService (`com.google.android.apps.actionsservice`)

EEA builds for regular Android devices upgrading to Android 12 or higher, if they previously preloaded `GoogleSpeechServices` app, it MUST be removed from the system image upon upgrade.

EEA builds for regular Android devices running Android 12 or higher, if they didn't preload the Google app, MUST preload the following headless apps:

- GoogleActionsService (`com.google.android.apps.actionsservice`)

End new requirements

Where the GMS Requirements contradict the GMS EEA Build requirements, the GMS EEA Build requirements take precedence.

Russia builds

GMS smartphones and tablets sold or distributed in the Russian Federation must meet the GMS Russia Build requirements (/gms/policies/restricted/russia-builds).

Where the GMS Requirements contradict the GMS Russia Build requirements, the GMS Russia Build requirements take precedence.

Turkey builds

GMS smartphones and tablets sold or distributed in Turkey under the optional Turkey MADA addendum must meet the GMS Requirements for Turkish Builds (/gms/policies/restricted/turkey-builds) and Google Search Widget Turkey Placement (/gms/policies/restricted/turkey-placement).

Where the GMS Requirements contradict the GMS Requirements for Turkish Builds for DEVICEs subject to the optional MADA addendum, the GMS Requirements for Turkish Builds take precedence.

6 Best Google Experience

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

6.1 Geo-availability

Each Google product included in GMS has a corresponding geo-availability definition which provides the list of countries that a Google product can be shipped into. It is subject to change over time depending on Google's product policies or regulatory compliance status. The latest geo-availability is available on the Geo-availability page (/gms/policies/overview/geo-availability).

[GMS-6.1-001] The software builds for a PRODUCT MUST be compliant with the geo-availability at the time the initial release (IR) build of the PRODUCT was approved by Google. For example, if a new PRODUCT is going to be on sale in the United States, its IR build MUST preload only the set of apps available for the United States at the time of approval, under the conditions set forth in the geo-availability definition.

[GMS-6.1-002] Partners MAY distribute the optional GMS apps listed in the geo-availability chart (/gms/policies/overview/geo-availability).

6.2 Core services and apps

Google releases the GMS bundle that includes not only all Core apps such as Chrome, Maps, or YouTube, but also all Core services such as GmsCore or ConfigUpdater. Integrating the GMS bundle preloads these Core services and apps in the system image.

[GMS-6.2-001] An IR build for a new PRODUCT submitted for approval MUST preload all Core services and apps in the latest GMS bundle no later than 60 days after it is released by Google. However, if Google has released a higher version of an individual Core service or app, partners MAY preload it instead of the one in the GMS bundle, as long as the software build complies with this document.

[GMS-6.2-002] MR builds are REQUIRED to update GMS apps if:

- A preloaded app must be updated to comply with the Security Patch Level declared by the build.
- The build upgrades the Android platform API level, and newer versions of GMS apps are required by the new platform.
- Google releases a fix for an issue that can be addressed only by updating the app in the system image.

[GMS-6.2-003] Partners SHOULD NOT include any GMS updates in security MR builds to ensure smaller incremental update sizes, unless a GMS module has to be updated to comply with the security patch level declared by the build.

Core services

[GMS-6.2-004] All Core services are listed below.

APK filename/package name	Description	Note
AndroidPlatformServices `com.google.android.gms.policy_sidecar_aps`	Google Services for Android Platform. This module was integrated in the Google Play services app in the past, but separated to allow independent updates.	**For DEVICEs running Android 9 or higher only.**
ConfigUpdater `com.google.android.configupdater`	Allows updates of nonexecutable system components over the air.	

GmsCore `com.google.android.gms`	Provides third-party apps with Google services API.	Also known as 'Google Play services'
GoogleBackupTransport `com.google.android.backuptransport`	Backup data to the user's Google account.	**MUST NOT be preloaded for DEVICEs running Android 10 or higher.**
GoogleFeedback `com.google.android.feedback`	Allows users to send bug reports or feedback to Google.	
GooglePartnerSetup `com.google.android.partnersetup`	Partner-specific client id initialization service.	
GoogleRestore `com.google.android.apps.restore`	Provides seamless device setup experience by restoring a device from the user's previous backup. This module was integrated part of Google Play services app in the past, but separated to allow independent updates.	**For DEVICEs running Android 9 or higher only.**
GoogleServicesFramework `com.google.android.gsf`	Remote service provisioning framework.	
SetupWizard `com.google.android.setupwizard`	Google's reference Setup Wizard implementation for the out-of-box setup user experience.	
GoogleContactsSyncAdapter `com.google.android.syncadapters.contacts`	Syncs contacts data to the user's Google account.	
WebViewGoogle `com.google.android.webview`	Renders web pages.	
TrichromeLibrary `com.google.android.trichromelibrary`	Common library for Chrome and WebView. See Chrome and WebView	**For DEVICEs**

	(#chrome-and-webview) for details.	**running Android 10 or higher only.**
GooglePackageInstaller `com.google.android.packageinstaller`	Provides runtime permission user interface.	
GoogleExtServices `com.google.android.ext.services`	Provides Android framework extension mechanism.	For DEVICEs running Android 10 or higher, this app is a Mainline module.
GoogleExtShared `com.google.android.ext.shared`	Same as above.	
GooglePrintRecommendationService `com.google.android.printservice.recommendation`	Allows users to discover and configure printers on the local network.	
AndroidAutoStub `com.google.android.projection.gearhead`	Stub app to enable Android Auto service on Android telephony devices.	**For DEVICEs running Android 10 or higher only.**
GoogleLocationHistory `com.google.android.gms.location.history`	Properly attributes location requests and provides visibility to users on location access.	**For DEVICEs running Android 10 or higher only.**

APK filename/package name	DescriptionNote	
GoogleTTS/Speech Services by Google `com.google.android.tts`	Provides text-to-speech	GoogleTTS (com.google.android.tts): Device running Android 11 or lower MUST preload t

	accessibility service.	app. This app MUST be replaced by Speech Services by Google when upgrading to Andr or higher. Speech Services by Google (`com.google.android.tts`): Devices running Android 12 or higher MUST preload this app app is a rebranded version of GoogleTTS.
GoogleCalendarSyncAdapter `com.google.android.syncadapters.calendar`	Syncs calendar data to the user's Google account.	SHOULD NOT be preloaded if standalone. G Calendar (https://play.google.com/store/apps/d id=com.google.android.calendar) app is preloaded. MUST use version 2021.23.3-389584493 or for Android 12 or higher. MUST use version 2020.20.5-319002662 or for Android 11 or lower.

JAR filename	Description
`com.google.android.maps.jar` `com.google.android.media.effects.jar`	These JAR files MUST be preloaded to support the Google APIs component in the Android SDK. For example, this component for Android 7.0 SDK is downloaded under `${ANDROID_HOME}/add-ons/addon-google_apis-google-24` folder. **Note:** `com.google.android.media.effects.jar` **MUST NOT be preloaded for DEVICEs running Android 9 or higher.** **Note:** `com.google.android.maps.jar` **MUST NOT be preloaded for DEVICEs running Android 11 or higher.**

Core apps (subject to the geo-availability)

- Google Play Store (https://play.google.com/store)
- Google Search (https://play.google.com/store/apps/details?id=com.google.android.googlequicksearchbox)
- Chrome browser (https://play.google.com/store/apps/details?id=com.android.chrome)
- Google Drive (https://play.google.com/store/apps/details?id=com.google.android.apps.docs)

- Gmail (https://play.google.com/store/apps/details?id=com.google.android.gm)
- Google Duo (https://play.google.com/store/apps/details?id=com.google.android.apps.tachyon)
- Maps (https://play.google.com/store/apps/details?id=com.google.android.apps.maps)
- YouTube Music (https://play.google.com/store/apps/details?id=com.google.android.music)
- Google Photos (https://play.google.com/store/apps/details?id=com.google.android.apps.photos)
- Google Play Movies and TV (https://play.google.com/store/apps/details?id=com.google.android.videos)
- YouTube (https://play.google.com/store/apps/details?id=com.google.android.youtube)

Note: On November 4, 2019, YouTube Music became one of Core apps, in place of Google Play Music.

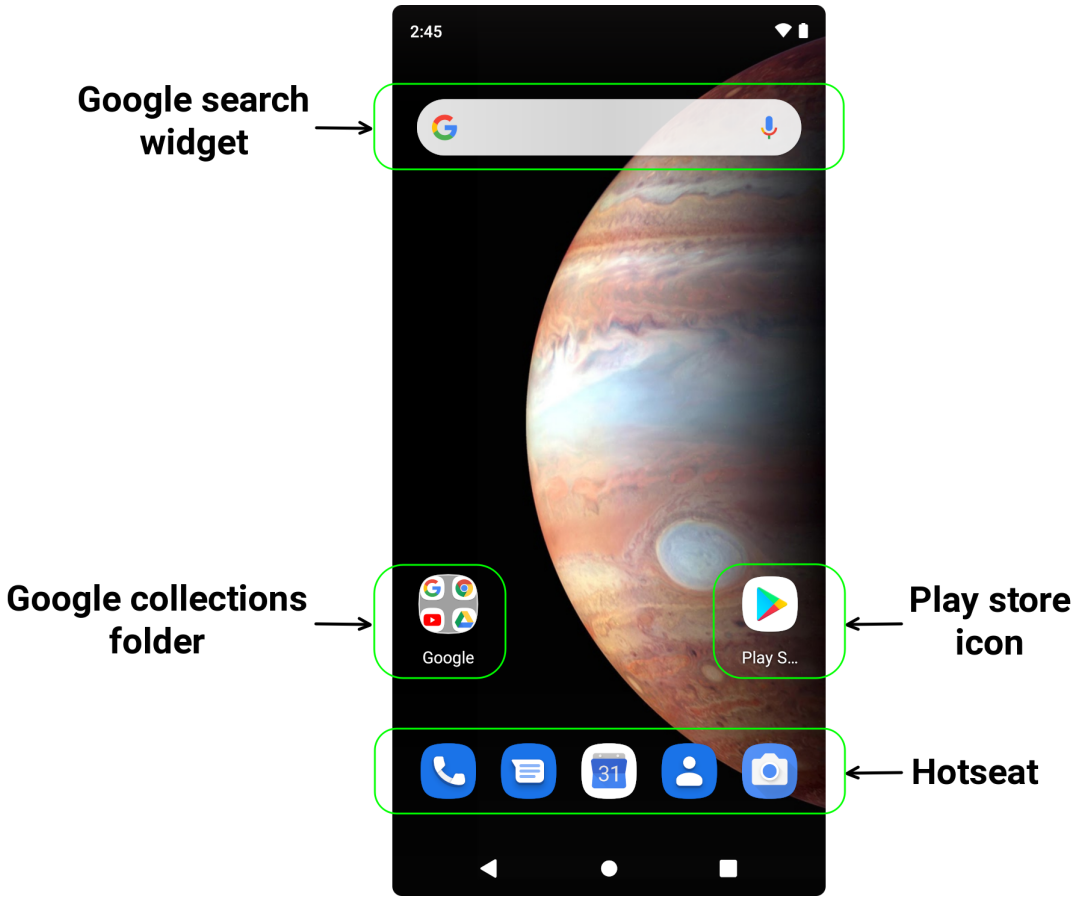
6.3 Default home screen layout

[GMS-6.3-001] The default home screen MUST be same as the one displayed by pressing the physical home button, tapping the home button on the navigation bar, or the home gesture.

[GMS-6.3-002] The default home screen MUST feature the following:

- Google Search widget
- Google collections folder containing Core app icons
- Play Store app icon

Here's an example layout for the default home screen.



Placement of the Google Search widget

To provide a consistent user experience, the approved implementation of the Google Search widget is the widget provided by Google Search app.

Note: The default preloaded launcher MUST NOT customize the Google Search widget. For example, the Launcher2 implementation in the AOSP provides a hardcoded Search widget, but it doesn't comply with this requirement. The latest Launcher3 implementation in the AOSP supports the Google Search widget in compliance with this requirement.

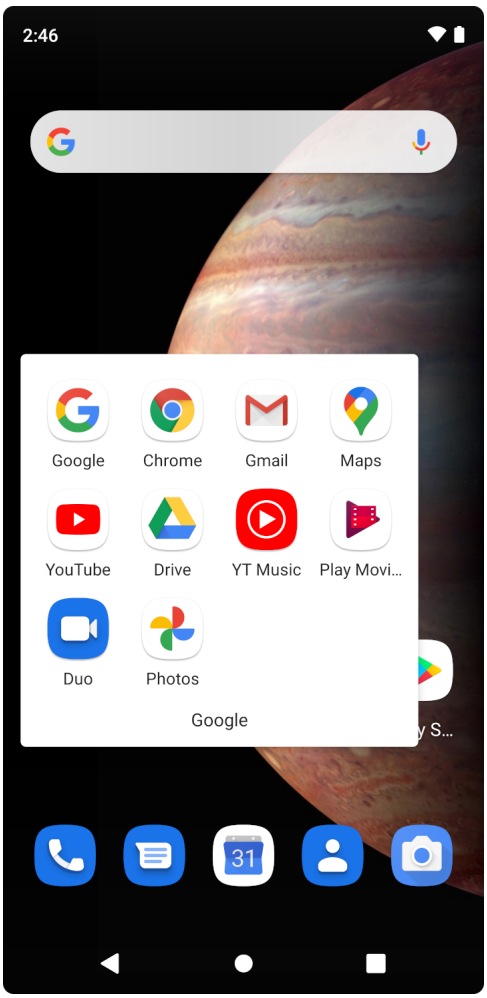
Placement of the Google collections folder

[GMS-6.3-004] The collections folder on the default home screen MUST:

- Be labeled *Google*.
- Have the Core app icons in the following order, left to right, top to bottom (subject to the geo-availability (/gms/policies/overview/geo-availability))
    1. Google Search

2. Chrome
3. Gmail
4. Maps
5. YouTube
6. Drive
7. YouTube Music
8. Play Movies
9. Duo
10. Photos

- NOT include any non-Google apps.



Placement of apps in hotseat or default home screen

[GMS-6.3-005] Any GMS app MAY alternatively be placed in the hotseat or default home screen. Core app icons MAY be removed from the Google collections folder if they're placed in the hotseat or default home screen.

6.4 Placement of the optional GMS apps

[GMS-6.4-001] Optional GMS app icons SHOULD adhere to one of the following placements:

- An icon on the default home screen
- The hotseat
- Within the Google folder, at the end of the collection (that is, after the Photos app)

6.5 Placement on Apps menu

Placement of GMS apps on the Apps menu is subject to the following requirements:

- **[GMS-6.5-001]** GMS apps MUST be placed in the Apps menu.
- **[GMS-6.5-002]** If a DEVICE implemented folders within the Apps menu, the Core app icons except for the Play Store SHOULD be placed within the Google folder.
- **[GMS-6.5-003]** Play Store SHOULD be in the top level of the Apps menu and MAY NOT reside within any folder on the Apps menu.

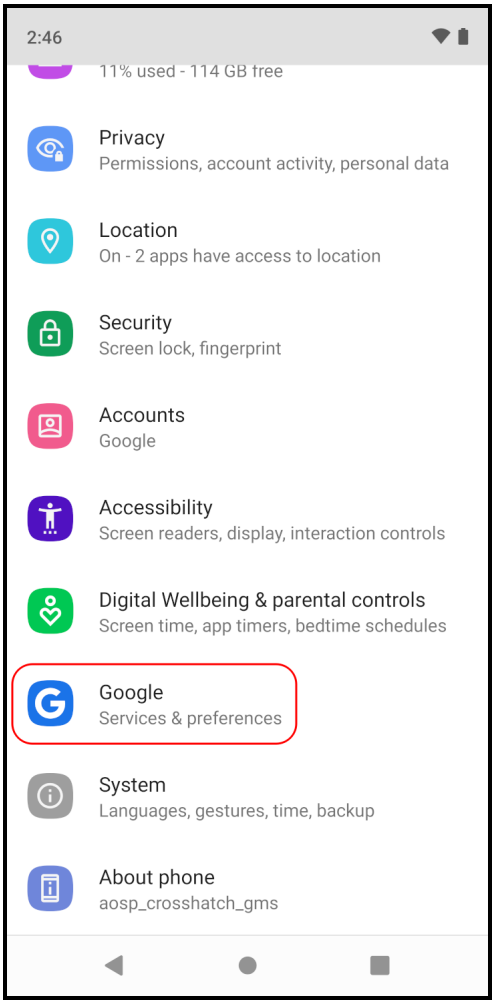
6.6 Privacy settings

[GMS-6.6-001] DEVICEs running Android 10 or higher MUST have these features:

- The Privacy section MUST be a top-level item in the Settings app.
- The Privacy section MUST feature the Google Privacy section injected from the Google Play services app.
- The Privacy page MUST also include a notification when accessibility services and/or when DEVICE management is active.

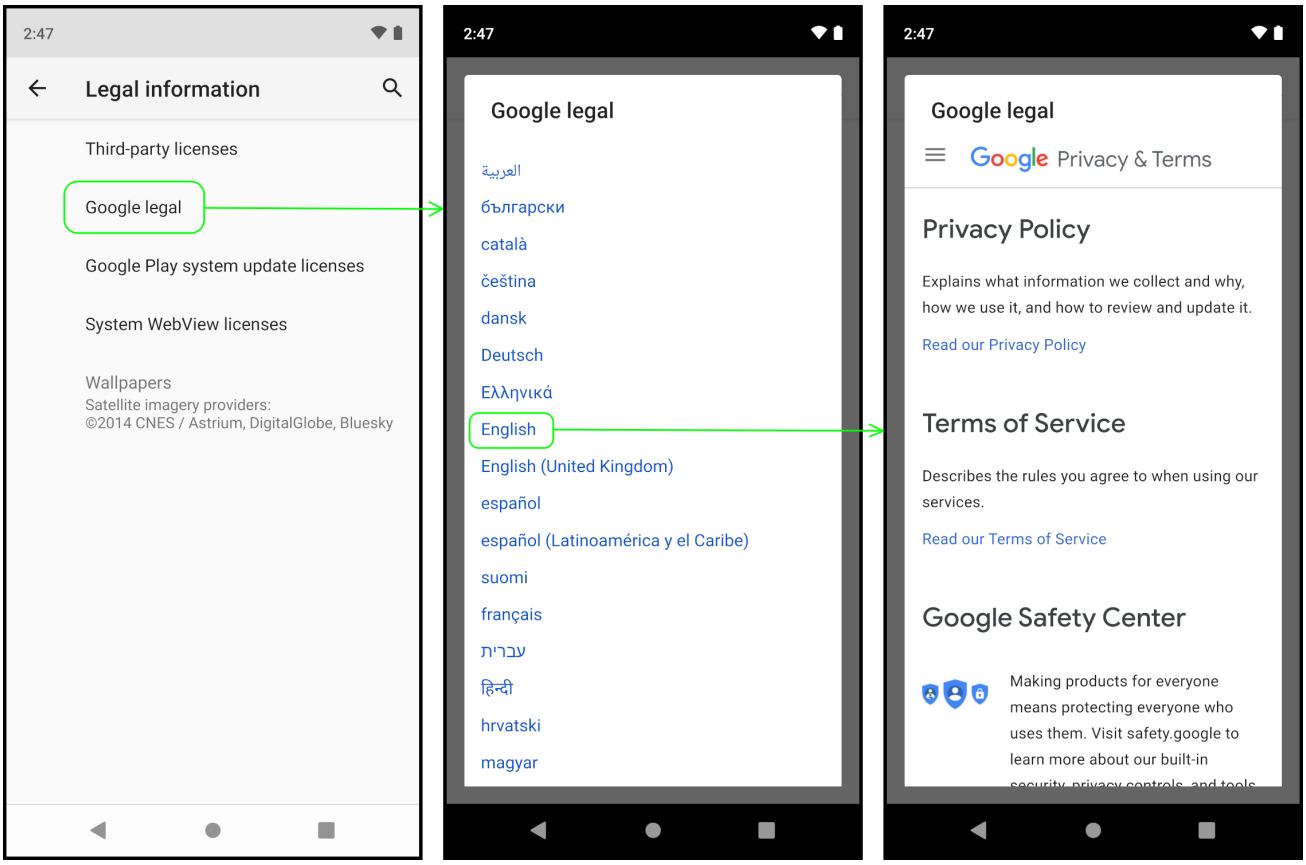
6.7 Google settings

[GMS-6.7-001] The AOSP Settings app allows other system apps to add extra top-level menu items to the settings UI, and the Google Play services app uses the feature to implement the Google Settings menu. When starting up, the Settings app collects extra menu items by searching for all system activities that resolve the intent `com.android.settings.action.EXTRA_SETTINGS` and `com.android.settings.action.IA_SETTINGS`. DEVICEs MUST leave this mechanism unchanged, so that it can present the top-level Google settings menu to the user.



6.8 Google legal settings

[GMS-6.8-001] The AOSP Settings app is designed to present a **Google legal** item under the Legal information settings. Tapping the Google legal menu item brings up a popup activity, which shows the Google legal terms page as shown in the following screenshots. This mechanism is implemented by querying an activity that resolves the `android.settings.TERMS` intent. DEVICEs MUST preserve this mechanism.



6.9 Backup transport configuration

[GMS-6.9-001] The GMS bundle provides the system configuration XML files and framework overlay files to allow Android platform to support GMS features correctly. DEVICEs MUST NOT modify the following backup configurations in the GMS bundle:

- Add Google Play services to the allowed list of the backup data transport
- Setting Google Play services as the default backup transport

6.10 Backup settings

[GMS-6.10-001] The AOSP Settings app implements the backup settings by calling the activity provided by the backup transport service. On DEVICEs running Android 8.0 or higher, Google backup settings MUST be accessible through this mechanism.

Note: Partners MAY add their own backup settings by putting an intent URI string to the `config_backup_settings_intent` resource, which is defined in `packages/apps/Settings/res/values/config.xml`.

6.11 Contacts metadata backup

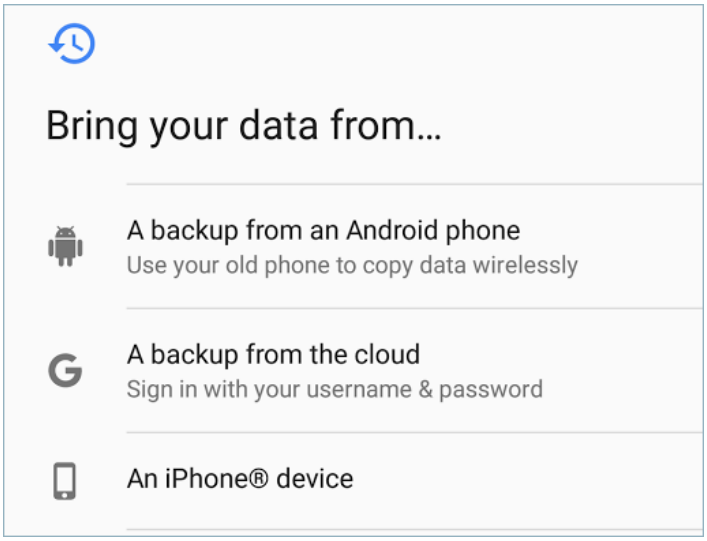
Android users customize their contacts on their device by choosing favorites or setting certain numbers or emails as primary emails for contacts, but such data is lost when users switch devices. Further, any information about how often a contact is called or texted is also lost, so the user's *Frequently Contacted* information is lost.

Android allows users to backup such contacts metadata in addition to contacts data so that they can get the exact contacts experience as they did on their old Android device, including usage stats, favorites, speed dial, custom merges, default photos, primary number, email settings, and more, when they get a new Android device.

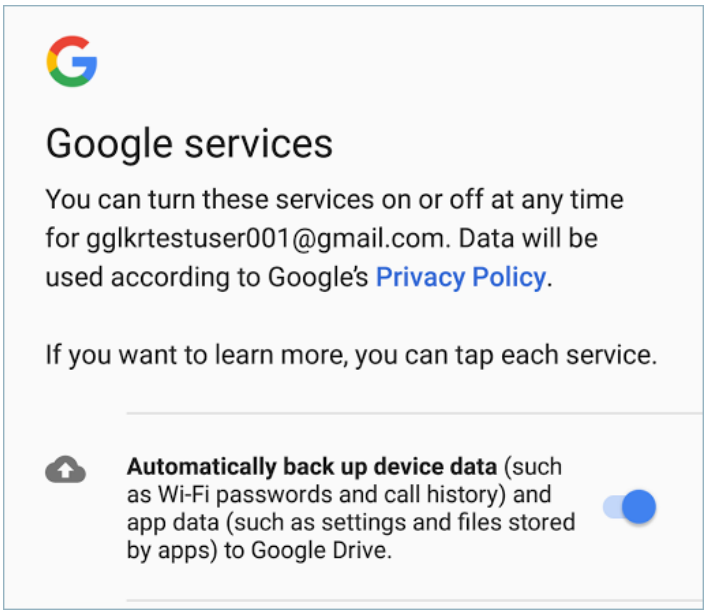
[GMS-6.11-001] The GMS bundle includes necessary framework overlay files to set Google Play services as the sync adapter for the information. DEVICEs MUST support this feature.

6.12 Restore experience in the Setup Wizard

[GMS-6.12-001] The Setup Wizard on DEVICEs is STRONGLY RECOMMENDED to use the reference device restore flow implemented in the GMS Setup Wizard app. Among the available restore options, a backup from the cloud flow in the following screenshot MUST be presented to the user unless the user has completed a device to device migration that covers a range of data that is the same or greater than the cloud flow. PRODUCTs launching with Android 9 and higher and presenting the option MUST present it before presenting the Google Account sign-in screen.



[GMS-6.12-002] If the user successfully signs in with a Google account, the Google services screen before the user exits the Setup Wizard. The backup service to the user's Google account MUST be provisioned if and only if the user consented to the backup option in the Google services screen.



[GMS-6.12-003] Restore experience in the Setup Wizard

Starting January 1, 2022, new IR build submissions, seeking GMS approval for DEVICEs launching with Android 12 or higher, MUST provide users with a user-skippable option in the Setup Wizard, which allows users to set up a device by copying apps and data from another device.

The option MUST be presented by the Setup Wizard.

This option, if it was skipped by users, MUST be presented again by the Deferred Setup.

Note: This requirement does not apply for Android Go devices.

[GMS-6.12-004] Restoring user data

The option in [GMS-6.12-003] MUST support all copying solutions below and let users choose one of them.

- Copying from another device running Android 5.0 or higher via USB cable
- Copying from another device running Android 5.0 or higher via Wi-Fi

- Copying from another non-Android device (Apple iPhone or iPad running iOS 12.0 or higher)

These solutions MUST support copying the following data from another device:

- Locally stored contacts
- Locally stored calendars (when available)
- Locally stored photos and videos

These solutions, if they copy from another Android device, MUST support copying the following data as well:

- SMS, MMS, RCS messages
- Apps
- Wallpaper
- Dark or light theme setting (when available)

Note: This requirement does not apply for Android Go devices.

6.13 Restoring settings

`SettingsBackupAgent` in Android 9 or higher is designed to handle any unexpected data in a robust manner when restoring system settings. It allows users to restore system settings reliably from a backup data that was created by a higher release of the Android platform.

[GMS-6.13-001] DEVICEs running Android 9 or higher MUST preserve this implementation behavior. For example, if any custom Partner extension was added to the AOSP `SettingsBackupAgent` to back up additional settings, the extension MUST validate the value of each setting at restore time because higher releases of Android might have introduced new values for the setting.

6.14 Android system metrics compatibility

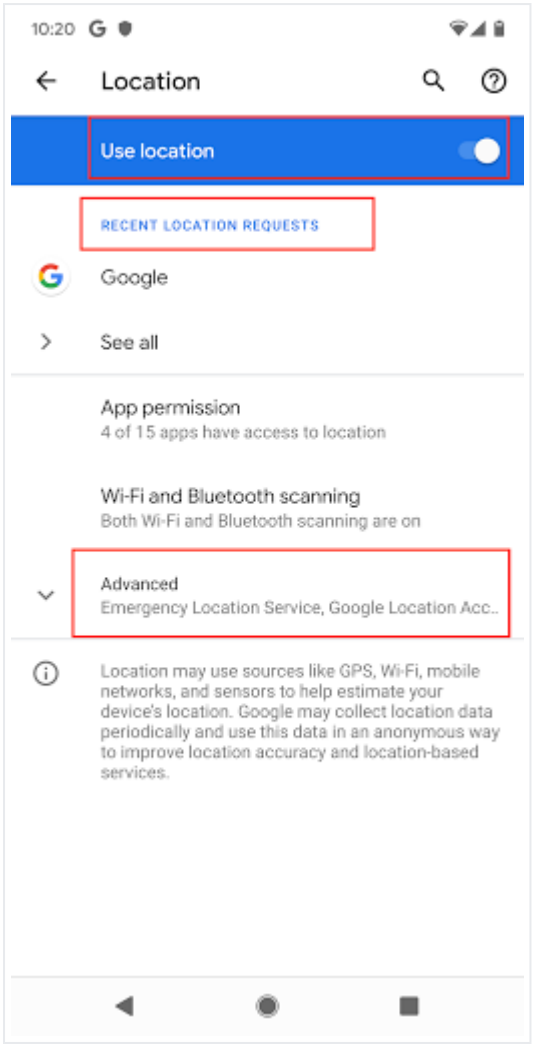
[GMS-6.14-001] DEVICEs running Android 10 or higher MUST maintain certain Android system metrics. Specifically, these DEVICEs MUST NOT change the purpose and intention

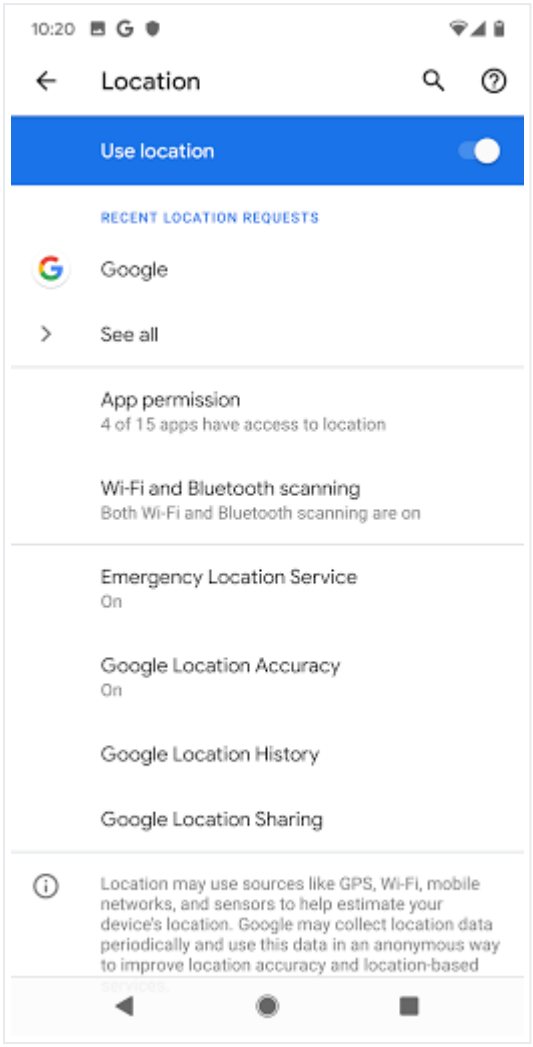
of the following data items collected by the AOSP statsd:

- `BinaryPushStateChanged`
- `SystemServerWatchdogOccurred`
- `ANROccurred`
- `AppCrashOccurred`
- `BatteryLevel`
- `WakeupAlarmOccurred`
- `WakelockStateChanged`
- `AppStartOccurred`
- `LmkKillOccurred`
- `SystemUptime`
- `ScreenStateChanged`
- `PluggedStateChanged`
- `TrainInfo`
- `WatchdogRollbackOccurred`
- `ProcessMemoryHighWaterMark`
- `TombStoneOccurred`

6.15 Location settings injection

[GMS-6.15-001] DEVICEs MUST preserve the mechanism in the AOSP Settings app that allows Google Play services app to inject settings items to the Location settings menu, as shown in the following screenshot.


10:20
Location
Use location
RECENT LOCATION REQUESTS
Google
See all
App permission
4 of 15 apps have access to location
Wi-Fi and Bluetooth scanning
Both Wi-Fi and Bluetooth scanning are on
Advanced
Emergency Location Service, Google Location Acc..
Location may use sources like GPS, Wi-Fi, mobile networks, and sensors to help estimate your device's location. Google may collect location data periodically and use this data in an anonymous way to improve location accuracy and location-based services.



6.16 Google location provider and activity recognizer

[GMS-6.16-001] DEVICEs MUST use the location providers (`NetworkLocationProvider` and `FusedLocationProvider`) implemented by Google Play services, and MUST NOT allow location provided by these providers to be repackaged and provided through another preloaded location provider or equivalent services to Android apps.

[GMS-6.16-002] DEVICEs running Android 12 or higher MUST configure the Google Play services app to hold the system Activity Recognizer role by adding its package name to the `config_systemActivityRecognizer` framework resource, and MUST NOT allow the activity information provided by the Google Play services to be repackaged and provided through another preloaded activity recognizer or equivalent services to Android apps.

6.17 CHRE/Context Hub requirements

Android Context Hub is an optional feature that is used by GMS to implement and/or enhance context-related features. The Context Hub Runtime Environment (CHRE) runs on a low-power processor and provides standard APIs for processing signals including sensors and location while the main apps processor is asleep.

[GMS-6.17-001] DEVICEs SHOULD support CHRE and the associated `ContextHubManager` system APIs. (Hereby "SHOULD" means "recommended" per IETF RFC 2119). DEVICEs that support CHRE and running on Android 11 or higher MUST declare the `android.hardware.context_hub` feature flag.

Note: Reach out to your TAM for the latest instructions customized to your hardware.

6.18 Nearby Share

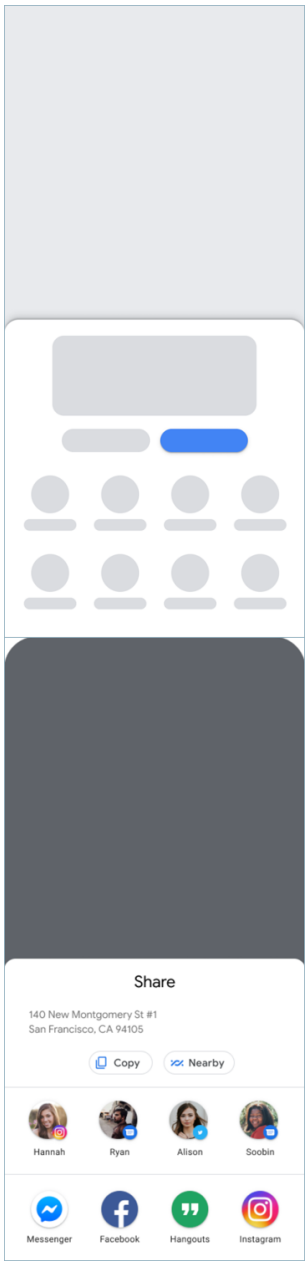
Nearby Share is a privacy-conscious, proximity-based, peer-to-peer sharing feature that lets users share a variety of things with other Android devices. Nearby Share involves a user interface integration in the following System UI elements:

[GMS-6.18-001] Starting January 11, 2021, PRODUCTs launching with Android 10 or higher, and PRODUCTs upgrading to Android 11 or higher MUST support the Nearby Share feature by integrating two UI elements as defined below and preloading the Google Play services app that supports the feature.

(1) Prominent entry point on Sharesheet can be satisfied through two options :

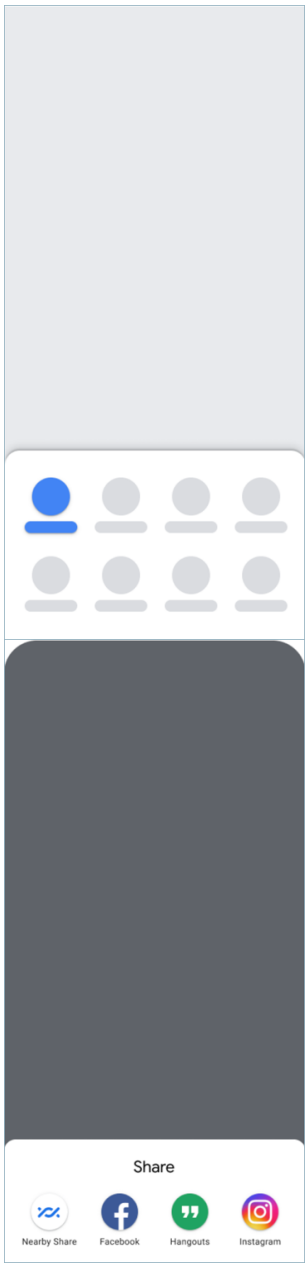
- (Recommended) Option 1: A visually differentiated Nearby Share entry point that must be placed above the list of recommended apps (for example, a chip in the action row of Android 11 AOSP Sharesheet).

  Example for (1): Action row entry point on Android 11 Sharesheet



- Option 2: A Nearby Share entry point in the first row or the first page of the Sharesheet, that's accessible by users without scrolling or swiping. This option is available ONLY IF the device doesn't support the visually differentiated implementation stated above.

  Example for (2): Pinning Nearby Share in recommended apps



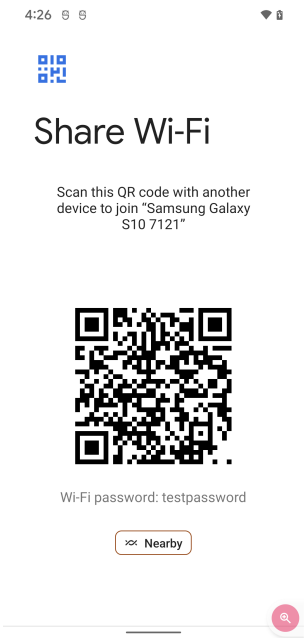
(2) Quick Settings tile must include (if the device implementation supports Quick Settings):

- A Nearby Share tile that is placed on the first or second page of the quick settings in the notification shade.

Note: To avoid adjusting user preferences, if a PRODUCT is upgrading to Android 11 or higher, placement of the Nearby Share quick settings tile is NOT REQUIRED. However, we STRONGLY RECOMMEND appending the Nearby Share tile to the last page so that users still have access to the tile.

[GMS-6.18-002] PRODUCTs launching with Android 12 or higher, and PRODUCTs upgrading to Android 12, if they include the Share Wi-Fi (via QR code) fragment in the Settings app,

MUST support sharing Wi-Fi configurations with nearby devices by placing an entry point to the feature in the fragment.



Refer to Nearby Share (/gms/policies/domains/nearby-share) for more information.

6.19 Call-based number verification

[GMS-6.19-001] Starting January 11, 2021, DEVICEs launching with Android 10 or higher, and DEVICEs upgrading to Android 11 or higher MUST enable the Google Play services app to access the call-based number verification API, by adding its package name to a resource overlay according to the integration guideline.

See Call-based Number Verification (/gms/building/connectivity/call-based) for more details.

6.20 Play as you download

On DEVICEs running Android 12 or higher:

- The **app content MUST NOT be read** except in support of running the app for a minimum of 2 hours after initiation of an installation of an APK on the Incremental file system (https://source.android.com/devices/architecture/kernel/incfs). Additional reads by the system or preloaded apps prevent the proper function of the Play as you download (https://developer.android.com/games/distribute/play-as-you-download) feature.

- The **launcher MUST show the app icon and the app MUST be launchable** while downloading as soon as the necessary portion of the app is downloaded.

It is **STRONGLY RECOMMENDED** to support the progress in app icons. If supporting the progress in app icons:

- **Show the app icon and download progress indicator**, such as a circular progress bar, on the home screen when an incremental app download is initiated from Google Play, while the app is still in downloading state and not ready to launch. Clicking the promise icon should take the user to the app details page in Google Play.
- **Show the regular app icon with download progress indicator** both on home screen and app drawer when a partially downloaded app is ready to be launched. The progress indicator should reflect the download state of the app. Clicking this icon should launch the app.
- **Remove the download progress indicator** from the app icon when the app is fully downloaded.

See Launcher Support for Incremental (/gms/policies/domains/incremental) and Creating Incremental Installer (/gms/policies/domains/incremental-installer) for more information.

7 App Specific

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Release	Requirements	Links to requirements
March 2022	7.1	Digital Wellbeing and parental controls

		(#ditigal-wellbeing-and-parental-controls)
March 2022	7.18	Google Kids Space (#google-kids-space)
March 2022	7.20	Entertainment Space (#GMS-7.20-001)
March 2022	7.21	Android System Intelligence (ASI) (#asi)
March 2022	7.22	Android Messages (#messages)

7.1 Digital Wellbeing and parental controls

Preload requirements

[GMS-7.1-001] PRODUCTs that launch with or upgrade to Android 9 or higher after September 3, 2019 MUST have a digital wellbeing solution with parental controls at the top level of the Settings app, either by integrating Google's Digital Wellbeing app (/gms/building/apps/productivity/digital-wellbeing) or developing a custom wellbeing and parental controls solution that complies with the feature requirements below.

Even if Google's Digital Wellbeing app isn't preloaded, these DEVICEs MUST:

- Preload Google Play services (GMS Core) version 17.7.85 or higher.
- Preserve `KID_POST_SETUP` as the final action in the Setup Wizard.
- Set the default supervision profile owner `config_defaultSupervisionProfileOwnerComponent` as GMS Core following the integration instructions (/gms/building/apps/productivity/digital-wellbeing) for Android 10.

Note: For Android PRODUCTs that ship with a preloaded digital wellbeing solution and are upgrading to Android 9 or higher, the feature requirements below MAY be granted an exception after a review by Google.

Feature requirements for OEM wellbeing solution

Note: OEMs that develop their own wellbeing solutions should expect these requirements to be updated and amended for each major Android platform release.

Note: OEMs MUST use the Google-provided translations (https://docs.google.com/spreadsheets/d/1SAL330RKp9Z7DcEKu0nhqQrkgAqaITKIe-8-Me_y3nM/preview) for requirements that include specific user-visible language.

The OEM wellbeing solution MUST include the features in this section of the GMS Requirements.

Settings integration

[GMS-7.1-002] The Settings app MUST have a top-level item titled **Digital Wellbeing & parental controls** or **Digital Wellbeing and parental controls**. This item MUST launch the OEM wellbeing solution.

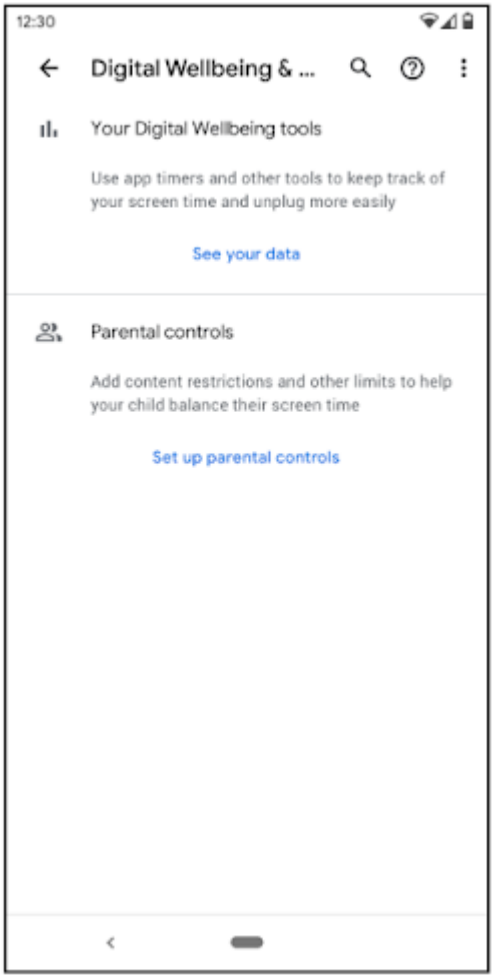
Parental controls integration

Start new requirements for 12

~~**[GMS-7.1-003]** When users launch Digital Wellbeing & parental controls for the first time, **DEVICEs MUST show an onboarding screen** where users can select between setting up parental controls for a child or using wellbeing features for themselves by viewing their usage dashboard. For example, the Pixel screen looks like this.~~

[GMS-7.1-003] When users launch Digital Wellbeing and parental controls, a Parental control entry point specifications (#parental-controls-entry-point) MUST be present in the OEM wellbeing solution to configure parental controls.

End new requirements



If the DEVICE is supervised with Family Link (/gms/building/apps/productivity/digital-wellbeing#check-supervision):

- Tapping the **Settings** menu item or launching the OEM wellbeing solution MUST include an intent to the parental controls component (/gms/building/apps/productivity/digital-wellbeing#launch-parental-controls) in Google Play Services.
- The OEM wellbeing usage dashboard and features MUST NOT be accessible, as this could cause confusion about which app applies restrictions.
- Any previously enabled Digital Wellbeing features MUST be disabled.

Parental control entry point specifications

[GMS-7.1-004] The entry point for parental controls MUST:

- Exist permanently in the OEM wellbeing solution.

Start new requirements for 12

- ~~Be visible on the onboarding screen without scrolling.~~

End new requirements

- Have a button or link labeled **Set up parental controls**.
- Include an intent to the Family Link parental controls component (/gms/building/apps/productivity/digital-wellbeing#launch-parental-controls) in Google Play Services.

If the entry point layout has a title, it MUST be **Parental controls**. If the entry point layout has a description, it MUST be *Add content restrictions and set other limits to help your child balance their screen time*.

Any differences or additional text used to describe the parental controls flow MUST be approved by Google.

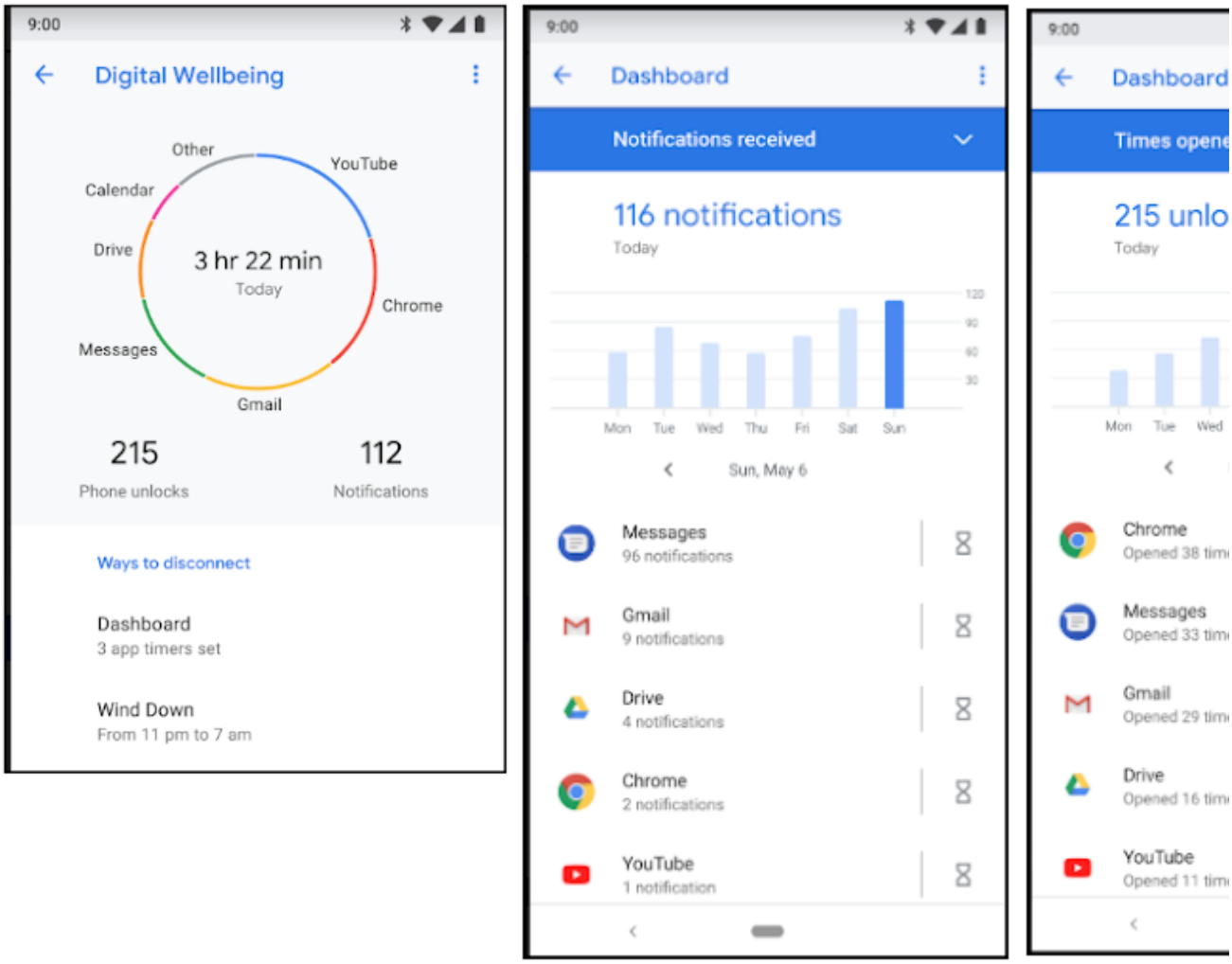
Usage dashboard

[GMS-7.1-005] The OEM wellbeing solution MUST provide a usage dashboard to the user that includes at least the following features:

- The wellbeing solution MUST provide a full DEVICE summary screen. The summary screen MUST provide metrics for the following DEVICE events, viewable to users for the current day and on a day-by-day basis:
  - Total amount of screen ON time
  - Number of DEVICE unlocks
  - Count of notifications received
- The wellbeing solution MUST support displaying app reporting with the following metrics, viewable to users on a day-by-day and hourly basis:
  - Total amount of screen ON time while the app is in the foreground
  - Number of times the app was opened
  - Count of notifications received
- The dashboard MUST retain history for at least 7 days.
- Users MUST be able to revoke app usage access to the wellbeing solution within the PRODUCT.

Use of the default launcher MUST NOT appear in the dashboard.

An example usage dashboard is shown in the following images.



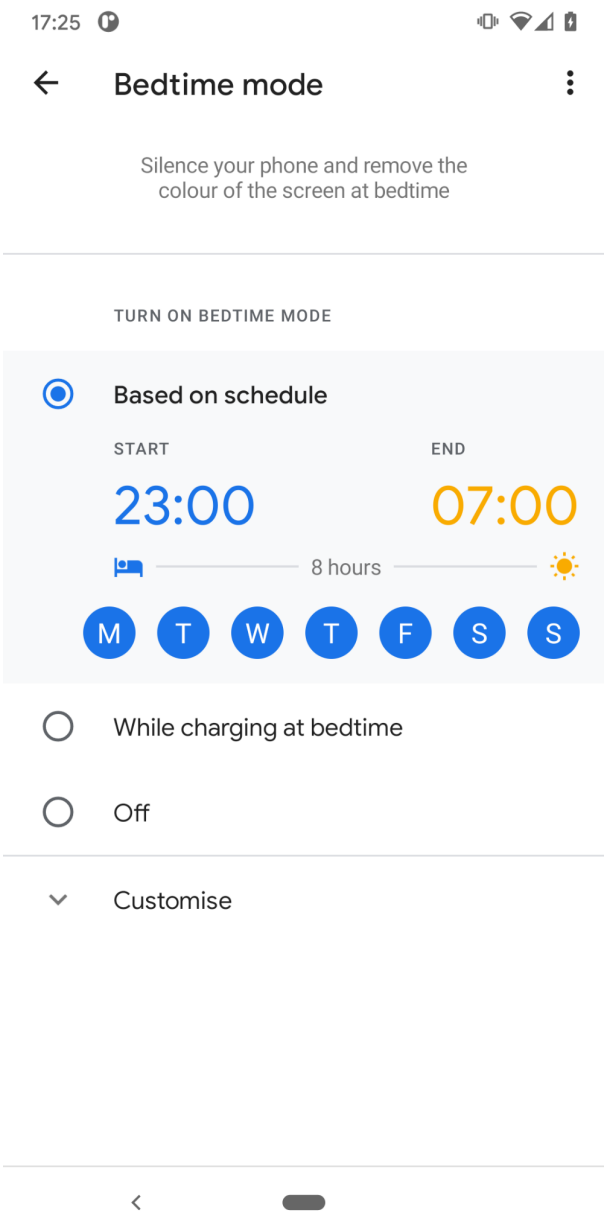
Daily app use limit

[GMS-7.1-006] The OEM wellbeing solution MUST provide the ability to set a daily use limit on a per-app basis. When users reach the app limit, notifications MUST be suspended for the remainder of the day. Important system apps and apps listed by `PackageManagerService#getUnsuspendablePackagesForUser` MUST NOT allow app limits.

Bedtime mode

[GMS-7.1-007.001] The OEM wellbeing solution MUST provide a feature that schedules Do Not Disturb (DND) and turns on grayscale. The DND feature MUST use Bedtime mode unless a partner has already implemented a digital wellbeing solution using Bedtime mode.

The solution MUST support configuring the schedule on a day-by-day basis. For example, users should be able to disable Bedtime mode over the weekend.



Optional features

[GMS-7.1-008] It's STRONGLY RECOMMENDED for the OEM wellbeing APK to support the following features with the recommended functionalities:

- Focus mode SHOULD provide a feature for users to limit app access. The app access limit feature SHOULD:
  - Be named **Focus mode**.
  - Provide an action in **Quick Settings**.
  - Disable notifications from the suspended apps.

    - Stop suspended apps from opening while **Focus mode** is on.
- Flip to Shhh SHOULD:
    - Let users turn on Do Not Disturb by placing the phone face down on a flat surface.
    - Vibrate when the phone enters and exits Do Not Disturb mode.
- Bedtime mode SHOULD:
    - Provide a persistent notification when enabled.
    - Match the end time of this feature to the alarm clock time by default.
    - Provide a **Quick Settings** tile.
- Daily app usage limits SHOULD:
    - Use the name **app timers**.
    - Suspend the app.
- Website timers for OEM browsers SHOULD:
    - Provide website suspension as part of wellbeing.
    - If the OEM wellbeing solution provides this functionality, website timers for OEM browsers MUST:
        - Provide hourly and daily use views.
        - Provide the option for users to opt out of this feature.

7.2 Usage stats event history

[GMS-7.2-001] UsageStatsManager MUST store and return at least 2 days of history of raw events. When storage permits, up to 8 days of history SHOULD be available. This lets wellbeing apps and predictive apps understand usage patterns and provide valuable suggestions to the user, and saves battery.

7.3 Android Auto

[GMS-7.3-001] DEVICEs launching or upgrading to Android 10 or higher that support `FEATURE_TELEPHONY` and return `false` for `ActivityManager.isLowRamDevice()` MUST

preload the Android Auto app as a privileged, headless Core service app in the system image. The Android Auto app MUST be granted Notification Listener access, and pregranted with the list of runtime permissions, as defined in Android Auto for Handheld Devices (/gms/building/apps/productivity/auto).

DEVICEs launching or upgrading to Android 12 or higher that don't support `FEATURE_TELEPHONY` or return `true` for `ActivityManager.isLowRamDevice()` (which includes Go edition devices) MUST NOT preload the Android Auto app in the system image, and are exempt from the additional Android Auto requirements below.

[GMS-7.3-002] DEVICEs launching or upgrading to Android 11 or higher are subject to [GMS-7.3-001] and MUST also label Android Auto and File Transfer together as the same option in any system-level USB configuration preference UI, for example, File transfer/Android Auto. Further, the system MUST mark this option as selected when USB is in either MTP or accessory mode (cf. `UsbDetailsFunctionsController.java`).

[GMS-7.3-003] DEVICEs launching or upgrading to Android 11 or higher, are subject to GMS-7.3-001 and MUST also honor multinetwork requirements (for example, Android CDD section 7.4.2 (https://source.android.com/compatibility/android-cdd#742_ieee_80211_wi-fi)) and Wi-Fi API contracts (for example, `WifiNetworkSpecifier`) without introducing additional OEM-specific UI that might otherwise interrupt wireless Android Auto use cases.

[GMS-7.3-004] DEVICEs launching with Android 11 or higher **before September 1, 2021** are STRONGLY RECOMMENDED to support 5 GHz Wi-Fi and clearly indicate such support in published product specifications, on relevant packaging and marketing materials, etc., for example, 802.11 a/b/g/n/ac (2.4 and 5 GHz).

[GMS-7.3-005] DEVICEs subject to [GMS-7.3-001] and launching with Android 11 or higher MUST support 5 GHz Wi-Fi, including short-range device (SRD) channels, and clearly indicate such support in published product specifications, on relevant packaging and marketing materials, and so on. For example, 802.11 a/b/g/n/ac (2.4 and 55 GHz).

Android Auto settings

[GMS-7.3-007] DEVICEs running Android 11 and preloading the Android Auto app MUST:

- Retain the **Android Auto** settings item under **Connected Devices** > **Connection Preferences** as implemented in the AOSP Settings app.
- Retain search behavior as implemented in AOSP Settings app, that is, show the Android Auto settings item in the search results for keywords declared by the Android Auto package's `SearchIndexablesProvider`.

OR

- Place the **Android Auto** settings item at the same level as **Bluetooth** settings, if there isn't a **Connected Devices** top-level settings item.
- If **Android Auto** isn't on the top level of settings and the settings page that leads to **Android Auto** has an item with summary text, then add *Android Auto* in the summary text of the top level item leading to **Android Auto**.
- Show the **Android Auto** settings item in the search results for keywords declared by the Android Auto package's `SearchIndexablesProvider`.

See Android Auto settings technical guide (/gms/building/apps/productivity/auto-settings) for implementation details.

USB port reset API

[GMS-7.3-008] DEVICEs launching with Android 12 or higher are STRONGLY RECOMMENDED to support USB port reset API.

7.4 Advanced audio framework changes

[GMS-7.4-001.001] DEVICEs launching with Android 10 or higher that have `FEATURE_TELEPHONY` AND Google Dialer app preloaded MUST support:

- Capture for `VOICE_UPLINK`, `VOICE_DOWNLINK`, and `VOICE_CALL` audio source
- A sink audio device of type `AudioDeviceInfo.TYPE_TELEPHONY`
- Directing playback for usage `AudioAttributes.USAGE_VOICE_COMMUNICATION` with a preferred device of type `AudioDeviceInfo.TYPE_TELEPHONY` to call TX path when in a call

[GMS-7.4-001.002] DEVICEs launching with Android 11 or higher with `FEATURE_TELEPHONY` are STRONGLY RECOMMENDED to support:

- Capture for `VOICE_UPLINK`, `VOICE_DOWNLINK`, and `VOICE_CALL` audio source
- A sink audio device of type `AudioDeviceInfo.TYPE_TELEPHONY`
- Directing playback for usage `AudioAttributes.USAGE_VOICE_COMMUNICATION` with a preferred device of type `AudioDeviceInfo.TYPE_TELEPHONY` to call TX path when in a call

Note: Implementation details can be found in Google Dialer Audio-based Features (/gms/building/apps/communication/audio-based).

7.5 Google Assistant

Setup experience

[GMS-7.5-001] A DEVICE MUST integrate the Google Assistant setup screens provided by the Google app with its Setup Wizard. Partners MAY customize the setup experience but MUST integrate at least two setup screens designated by Google.

However, if the Google Assistant doesn't support any of the primary languages of the market that the DEVICE targets, a waiver can be granted. The list of languages supported by Google Assistant is available from the Google Assistant Help Center (https://support.google.com/assistant/answer/7394513).

Basic actions

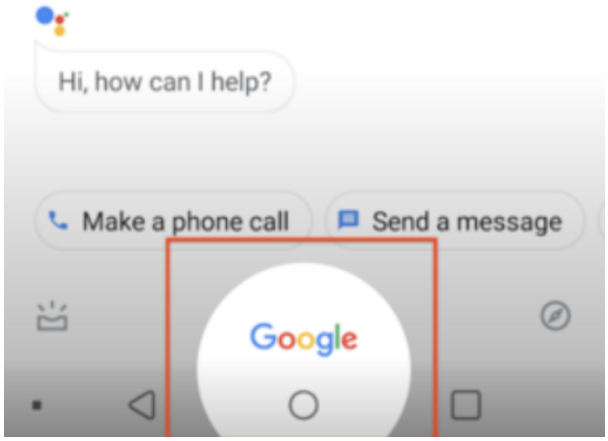
[GMS-7.5-002] Google Assistant supports the basic actions such as *Take a picture*, which are implemented using publicly available Android APIs and intents. DEVICEs MUST support all basic actions listed in Basic Actions (/gms/building/apps/assistant/basic).

Edge flashing animation and white orb animation

[GMS-7.5-003] From September 3, 2019 onward, all new DEVICEs MUST:

- Implement the **Flash Screen** setting to indicate to the end user that the context is shared with the Google Assistant, as defined in CDD section 3.8.4., C-2-1 (https://source.android.com/compatibility/android-cdd#384_assist_apis).
- Disable the **White Orb** animation effect (in red rectangle in the image below) shown when a user launches the default Assistant app.
- Disable the **White Orb** animation effect (in red rectangle in the image below) shown when a user launches the default Assistant app.

  The White orb animation is shown below.



- Have the out-of-box setting **Off** by default.

7.6 Google Location History

[GMS-7.6-001] The package name of the Google Location History app (`com.google.android.gms.location.history`) MUST be added to the `config_locationExtraPackageNames` framework resource overlay.

7.7 Chrome and WebView

[GMS-7.7-001] The GMS bundle for Android 7.0, 8.0, 8.1, and 9 releases include the **Monochrome** release of the Chrome app that can function as both a standalone Chrome browser and WebView provider. When Android 7.0, 8.0, 8.1, and 9 DEVICEs preload the Chrome app, they can also preload the WebView stub app in place of a fully functioning WebView app to reduce the size of the system image. However, the fully functioning WebView app MUST be preloaded if the type of DEVICE or the Geo-Availability allows a DEVICE configuration without the Chrome app on the system image.

[GMS-7.7-002] DEVICEs launching or upgrading to Android 10 or higher MUST preload the **Trichrome** release of Chrome and WebView apps. In addition, a new headless mandatory TrichromeLibrary app MUST also be preloaded, because both Chrome and WebView rely on it. The version of preloaded Chrome, WebView, and TrichromeLibrary APK files in the system image MUST exactly match.

Note: If a DEVICE must preload WebView without a standalone Chrome browser by the Geo-Availability or regulatory requirements, it MUST preload WebView and TrichromeLibrary.

For the detailed requirements, refer to Integrating WebView and Chrome (/gms/building/apps/core/webview-chrome).

[GMS-7.7-003] DEVICEs MUST provide a menu in the Settings app that lets users switch between available WebView providers on the DEVICE. Its user experience MUST align with the AOSP implementation.

7.8 Widevine DRM

[GMS-7.8-001] Widevine DRM is implemented with the OEMCrypto API library provided by the SoC or OEM partners. DEVICEs MUST support a certain version of the OEMCrypto API depending on the Android platform releases and the conditions below:

- Android 11: Version 16 REQUIRED for new DEVICEs. Version 16 STRONGLY RECOMMENDED for upgrading DEVICEs with secure AV1 support. Version 15 or 16 for other upgrading DEVICEs.
- Android 10: Version 15 for new DEVICEs, version 14 or 15 for upgrading DEVICEs
- Android 9: Version 14 for new DEVICEs, version 13 or 14 for upgrading DEVICEs
- Android 8.1: Version 13 for new DEVICEs, version 11 for upgrading DEVICEs
- Android 8.0: Version 11

[GMS-7.8-002] Android 12: Version 16 is REQUIRED for new DEVICEs. Version 16 is STRONGLY RECOMMENDED for upgrading DEVICEs with secure AV1 support. Version 15 or 16 is required for DEVICEs upgrading to android.hardware.drm@1.3 or android.hardware.drm@1.4.

Note: Android 8.0 or higher DEVICEs MUST NOT implement Widevine classic.

7.9 YouTube requirements

[GMS-7.9-001] DEVICEs launching Android 9 or higher are STRONGLY RECOMMENDED to support video decoding profiles, as described in Video Decoding (/gms/building/settings/overview/media-codec).

[GMS-7.9-002] DEVICEs launching Android 10 or higher MUST support simultaneous encrypted and unencrypted decoder sessions. The encryption level MUST match the highest encryption level that the DEVICE supports.

7.10 Instant apps

[GMS-7.10-001] Android 8.0 or higher DEVICEs MUST configure the Google Play services app as `EphemeralResolverService`, by overriding the value of the framework configuration resource `config_ephemeralResolverPackage` that is defined in `frameworks/base/core/res/res/values/config.xml`.

7.11 Google Factory Reset Protection

GMS provides a reference FRP implementation that meets the Factory Reset Protection. For the detailed requirements and integration instructions, refer to Configuring Factory Reset Protection (/gms/building/settings/security/configuring-factory-reset).

[GMS-7.11-001] DEVICEs supporting Google FRP MUST allocate at least 500 KB of space to the partition that backs `PersistentDataBlockService`.

7.12 Hotword library version API

[GMS-7.12-001] For DEVICEs launching with Android 11 that have a DSP running the Google Hotword Engine, the `SoundTrigger.ModuleProperties` object returned by the `SoundTriggerManager#getModuleProperties` system API MUST reflect the Google hotword library version running on the DSP firmware and a nonempty unique identifier describing the DSP platform. The HAL implementation MUST populate the `supportedModelArch` field by requesting those from the Google DSP hotword library directly.

[GMS-7.12-002] DEVICEs launching with Android 12 that have a DSP running the Google Hotword Engine:

- The `com.google.android.googlequicksearchbox` package MUST be granted the `SOUND_TRIGGER_RUN_IN_BATTERY_SAVER` privileged permission
- The `BatterySaverPolicy` for `ServiceType.SOUND` MUST be `SOUND_TRIGGER_MODE_CRITICAL_ONLY` or `SOUND_TRIGGER_MODE_ALL_ENABLED` with `SOUND_TRIGGER_MODE_CRITICAL_ONLY` being the preferred option.

Note: See Hotword DSP Platform Identifiers (/gms/building/apps/assistant/hotword-dsp) for implementation details.

7.13 DND access for GMS Core

[GMS-7.13-001] DEVICEs running Android 11 or higher MUST pregrant DND access to the Google Play services (by adding the package name to `config_defaultDndAccessPackages`).

7.14 Google Lens integration

Google Lens integration runs within the Android Google Search App (AGSA), which is Google's main app on Android. Lens SDK is available for download on the Google Lens page. Lens SDK and logo are available for download at Google Lens SDK (https://drive.google.com/a/google.com/uc?export=download&id=1Zc2cxA8QX7de8QJSC_6YOFGFfqWXbSM5) and Google Lens Branding (https://drive.google.com/a/google.com/uc?export=download&id=1TnRv7poRfwbQvWAA8M8n0gqplG--1dsL), respectively. The Lens SDK needs to be integrated following the Google Lens integration guidelines (/gms/building/apps/lens#integration-steps).

The Lens libraries that provide extended functionality to AOSP assume an Android-compatible DEVICE, which allows maximum compatibility with the Android app development ecosystem and distribution systems. DEVICEs MAY optionally integrate Google Lens in their app. However, DEVICEs integrating the Google Lens feature MUST meet the requirements below.

[GMS-7.14-001] An Android DEVICE with the following hardware features is REQUIRED:

- A rear-facing camera (as described in CDD section 7.5.1 (https://source.android.com/compatibility/android-cdd#751_rear-facing_camera))

The DEVICE MUST make this feature available to third-party apps.

[GMS-7.14-002] At least one rear-facing color camera on the DEVICE MUST support `CameraCharacteristics.INFO_SUPPORTED_HARDWARE_LEVEL_LIMITED` or higher.

[GMS-7.14-003] At least one rear-facing color camera on the DEVICE MUST be configurable with standard Camera2 APIs, or MUST have default values that support acceptable Google

Lens performance on the DEVICE:

- `CaptureRequest.CONTROL_AF_MODE`
- `CaptureRequest.CONTROL_VIDEO_STABILIZATION_MODE`
- `CaptureRequest.EDGE_MODE`
- `CaptureRequest.FLASH_MODE`
- `CaptureRequest.LENS_OPTICAL_STABILIZATION_MODE`
- `CaptureRequest.NOISE_REDUCTION_MODE`

[GMS-7.14-004] At least one rear-facing color camera on the DEVICE SHOULD conform to the following specification:

- Simultaneously supports at least one 1920x1080 SurfaceTexture stream and one 640x480 stream

[GMS-7.14-005] At least one rear-facing color camera on the DEVICE SHOULD support the following features:

- Zoom
- Flash

[GMS-7.14-006] At a minimum, the DEVICE main processor is STRONGLY RECOMMENDED meet either of the following requirements:

- A main processor with at least eight (A53 class) cores of at least 1.4 GHz that allow simultaneous execution of prioritized processing threads
- A main processor with at least two (A57 class) cores of at least 1.9 GHz, and two (A53 class) cores of at least 1.6 GHz

[GMS-7.14-007] At a minimum, the DEVICE memory MUST meet the following requirement:

- More than 2 GB of RAM

[GMS-7.14-008] At a minimum, the DEVICE screen MUST meet the following requirement:

- 720 px screen width

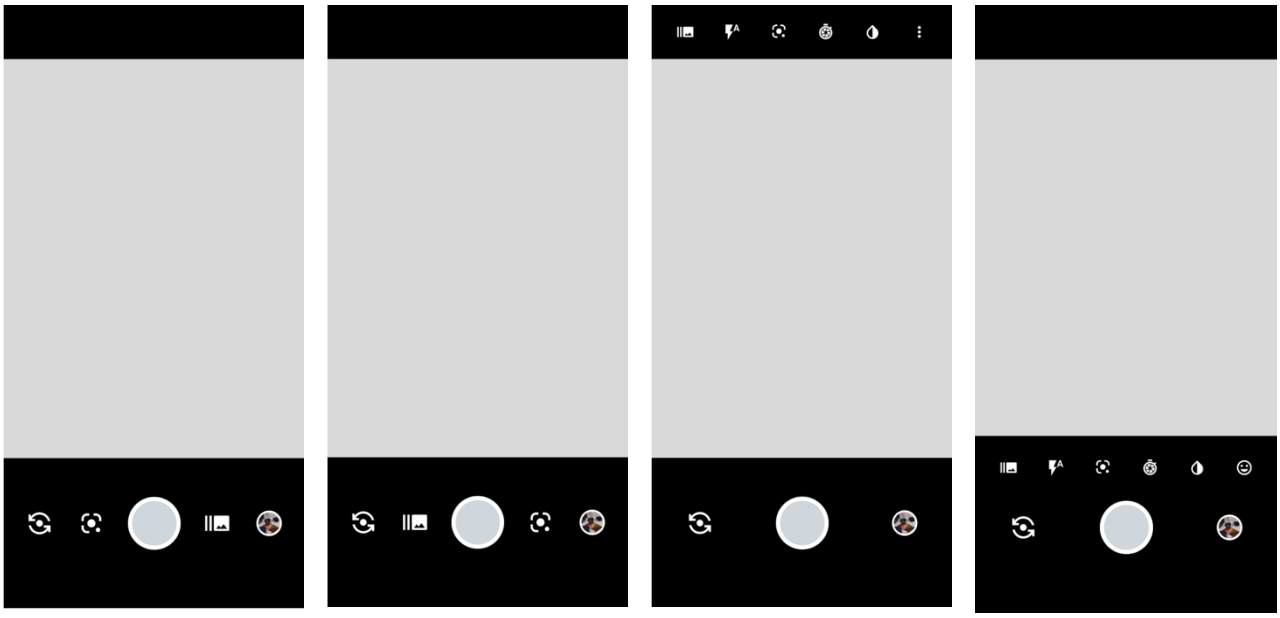
[GMS-7.14-009]

- The app that the Google Lens Integration SDK is integrated into MUST be the system's default camera app.

- The package name of the app that the Google Lens Integration SDK is integrated into MUST be identified in the `ro.com.google.lens.oem_camera_package` system property.

[GMS-7.14-010] It's STRONGLY RECOMMENDED to adhere to the placement requirements of Google Lens icon as described in the Google Lens branding guidelines (/gms/building/apps/lens/branding-guidelines).

[GMS-7.14-011] After opening the native camera app on a DEVICE with Google Lens SDK integrated, Google Lens should be featured as prominently as the video mode to the end user in terms of form, fit, and function.



[GMS-7.14-012] Google Lens integration doesn't support large screen (tablet) DEVICEs. Google Lens integration MUST NOT be integrated on DEVICEs where `Configuration.smallestScreenWidthDp` is greater than or equal to 600 dp.

[GMS-7.14-013] Google Lens integration requires the support of AGSA version 11.33 or higher.

[GMS-7.14-014] The `com.google.lens.feature.CAMERA_INTEGRATION` feature flag MUST be declared as described in the Google Lens integration guidelines (/gms/building/apps/lens#integration-steps).

[GMS-7.14-015] Google Lens image integration MUST support of AGSA version 12.24 or higher.

[GMS-7.14-016] The `com.google.lens.feature.IMAGE_INTEGRATION` feature flag MUST be declared as described in the Google Lens integration guidelines (/gms/building/apps/lens#integration-steps).

[GMS-7.14-017] The package name of the app that has Google Lens image integration MUST be identified in the `ro.com.google.lens.oem_image_package` system property.

7.15 Android Device Lock

[GMS-7.15-001] DEVICEs launching on Android 11 or higher on or after July 5, 2021 that support the Android Device Lock solution MUST not preload the Device Lock Controller APK `com.google.android.apps.devicelock`.

7.16 Accessibility

[GMS-7.16-001] DEVICEs launching with Android 12 or higher **MUST** provide following accessibility services out of box:

- The **screen reader** service for visually impaired users, which enables users to navigate the DEVICE without seeing the screen by having screen content read back to them.
- The Android Accessibility Suite app (`talkback.apk`) in the GMS provides the **screen reader** feature. Partners may preload their own screen reader implementation but it must implement the full DEVICE navigation and control capabilities.

[GMS-7.16-002] DEVICEs launching with Android 12 or higher are **STRONGLY RECOMMENDED** to provide following accessibility services out of box:

- The **switch access** service for users with severe motor impairments, which needs to enable users to navigate the DEVICE without touching the screen, using only one or more switches.

See Android Accessibility Suite (/gms/building/settings/accessibility/android-accessibility-suite) for more information.

7.17 Files by Google

[GMS-7.17-001] On GMS DEVICEs preloading the Files by Google app in the system image, the launcher icon of DocumentsUI MUST be hidden.

[GMS-7.17-002] Non-GMS DEVICEs preloading the Files by Google app in the system image MUST use the China version of the Files by Google app.

See Files by Google (/gms/policies/domains/files-by-google) for instructions on partner setup and how to hide the file icon.

7.18 Google Kids Space

Google Kids Space is a tablet experience with content to help kids discover, create, and grow. When Google Kids Space is preinstalled, parents have the option to set it up as a default launcher for their kids.

[GMS-7.18-001] DEVICEs preloading Google Kids Space MUST:

- Install only on tablets (DEVICEs with 7-inch or larger screens) running on Android 10 or higher.
- Install only in the system image.
- Have at least 2 GB of RAM.
- Enable multiuser support (https://source.android.com/devices/tech/admin/multi-user).
- Meet the requirements detailed in Google Kids Space integration (/gms/policies/domains/google-kids).

Start new requirements for 12

[GMS-7.18-002] ALL GMS tablet DEVICEs MAY provide users with an alternative out-of-box system launcher, which implements a **child-friendly home experience** (for example, the Google Kids Space app).

The **child-friendly home experience** launcher is exempted from the requirement GMS-6.3-002 (/gms/policies/domains/best-google-experience#GMS-6.3-002) (placement of the Google Search widget, the Play Store icon, and the Google collections folder on DHS) if it meets all of these conditions:

- MUST be used only in Android *tablet* DEVICEs as defined in CDD (https://source.android.com/compatibility/12/android-12-cdd#26_tablet_requirements).
- MUST be enabled out-of-box only when the DEVICE has been activated as a **supervised device** by Google Family Link.
- MUST NOT feature an app store icon or a search widget on its default home screen.

End new requirements

7.19 Speech Services by Google

[GMS-7.19-001] DEVICEs running Android 12 or higher MUST set the Speech Services by Google app as the system default speech recognizer by setting its package name `com.google.android.tts` to the framework overlay `config_systemSpeechRecognizer.` `<string name="config_systemSpeechRecognizer translatable="false">` `com.google.android.tts</string>`.

Start new requirements for 12

7.20 Entertainment Space

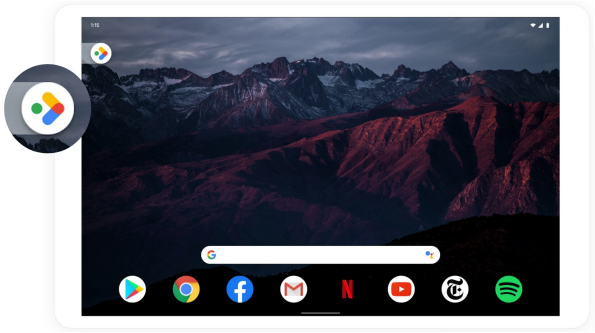
Entertainment Space is a tablet experience that helps users discover entertainment, such as videos, games, music, and books, right from the device launcher.

[GMS-7.20-001] DEVICEs preloading Entertainment Space:

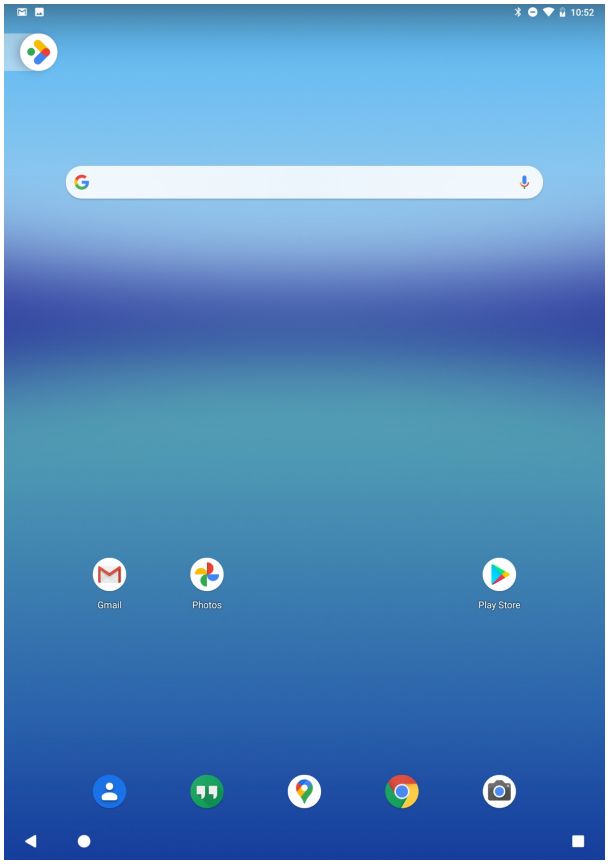
- MUST install only on tablets (DEVICEs with 7-inch or larger screens) running on Android 6 or higher.
- MUST be preloaded as a privileged app in the system image, by placing it in the folders like `/system/priv-app` or `/product/priv-app`.
- MUST be installed on a tablet with at least 2 GB of RAM.
- MUST allowlist (https://source.android.com/devices/tech/config/perms-whitelist) the following privileged permissions:
  - `android.permission.PACKAGE_USAGE_STATS`
  - `android.permission.ACCESS_NETWORK_STATE`
  - `android.permission.GET_ACCOUNTS` (dangerous)
  - `android.permission.GET_PACKAGE_SIZE`
  - `android.permission.INTERNET`
  - `android.permission.RECEIVE_BOOT_COMPLETED`
  - `android.permission.QUERY_ALL_PACKAGES`
  - `com.google.android.providers.gsf.permission.READ_GSERVICES`
- MUST preinstall the following APKs on the DEVICE.

    - **Youtube** minimum version: 5.36.5
    - **Google Play Books** minimum version: 5.12.5
    - **Google TV** (or **Google Play Movies & TV**, as applicable for the country) minimum version: 4.26.22
    - **GMS Core** minimum version: 20.39.15
        - **GMS Core Go** (instead of GMS Core) minimum version: 20.39.15 for low-RAM DEVICEs
    - **Play Store** minimum version: 22.1.18
- MUST have the new DEVICE default state for Entertainment Space enabled.
- MUST integrate Entertainment Space with the DEVICE's Launcher app using the provided `LauncherClient` AAR (Android library) (https://developer.android.com/studio/projects/android-library).
    - Entertainment Space MUST replace the entire -1 screen (swipe to the right) experience for users on integrated DEVICEs as described in the Integrating Entertainment Space (/gms/building/apps/productivity/ent-space) guide.
    - MUST use a Peeky Tab (/gms/building/apps/productivity/ent-space#peekytab) button with the Entertainment Space logo on the home screen to indicate to the user that Entertainment Space is present in the -1 screen.
        - UX requirements
            - The Peeky tab, the Entertainment Space logo on the home screen, MUST be on the top left for left to right languages (and top right for the right to left languages).
            - Peeky tab size
                - On 7-inch and smaller tablets:
                    - MUST not add any extra spacing around the icon.
                    - MUST set icon size to 48 dp (minimum requirements of A11Y).
                - On tablets larger than 7 inches:
                    - MUST set 24 dp padding to the left (right for RTL languages) of the icon.
                    - MUST set icon size to 56 dp.

- MUST place Entertainment Space Peeky tab without colliding with the Quick search bar widget.
    - If there is enough space to host Peeky tab above the search bar (68 dp margin + (56 dp or 48 dp height of the Peeky tab)), Peeky tab MUST be displayed above the search bar.



    - If there isn't enough space to host the Peeky tab above the search bar, the Peeky tab MUST be displayed below the search bar container.



    - The Peeky tab icon MUST be added in a way that the Peeky tab icon slides with the entire -1 screen when the user swipes into the -1 screen.

Integration guide

See Entertainment Space (/gms/building/apps/productivity/ent-space) for the integration guide.

7.21 Android System Intelligence

Android System Intelligence (ASI) provides privacy-preserving intelligence capabilities protected by Private Compute Core. Starting from Android 12, two groups of ASI capabilities are provided as the following GMS APK files for preload.

Private Infrastructure APK provides a group of private-by-default infrastructure implementations and associated APIs to DEVICE manufacturers and app developers. This APK includes the following capabilities:

- Speech Recognition Service
- Text classifier
- Keyboard suggestions

Private Features APK provides a group of novel intelligence capabilities to users. Currently, this APK includes the following capabilities:

- All capabilities in the Private Infrastructure APK
- App suggestions
- Live caption (starting Android T)
- Screen attention

Note: Private Infrastructure APK and Private Features APK share the same package name `com.google.android.as`.

Private Infrastructure

[GMS-7.21-001] DEVICEs launching or upgrading to Android 12, if they have 4 GB or more RAM, are STRONGLY RECOMMENDED to preload the Private Infrastructure APK.

Private features

[GMS-7.21-003] DEVICEs launching or upgrading to Android 12 or higher MAY optionally preload the Private Features APK in place of the Private Infrastructure APK. DEVICEs

preloading the Private Features APK are considered to comply with [GMS-7.21-001] or [GMS-7.21-002].

[GMS-7.21-004] DEVICEs preloading the Private Infrastructure APK MUST meet the following integration requirements.

- A system feature string `com.google.android.feature.ASI_MINIMAL` MUST be declared.
- A system feature string `com.google.android.feature.ASI` MUST NOT be declared.
- A system feature string `com.google.android.feature.DPS` (legacy ASI feature) MUST NOT be declared.

[GMS-7.21-005] DEVICEs preloading the ASI APK (the Private Infrastructure APK or the Private Features APK) MUST meet the following integration requirements.

- The Private Compute Services APK (`com.google.android.as.oss`) MUST be preloaded along with the ASI APK, as a privileged app in the system image.
- The ASI APK MUST be preloaded as a privileged app in the system image and allowlisted to use the following privileged permissions.
  - `android.permission.READ_OEM_UNLOCK_STATE`
  - `android.permission.WRITE_SECURE_SETTINGS`
- The ASI APK must be configured as the system role holder of the following roles.
  - `SYSTEM_AUDIO_INTELLIGENCE` (`config_systemAudioIntelligence`)
  - `SYSTEM_NOTIFICATION_INTELLIGENCE` (`config_systemNotificationIntelligence`)
  - `SYSTEM_TEXT_INTELLIGENCE` (`config_systemTextIntelligence`)
- The ASI APK and the Private Computer Services APK MUST NOT be allowed to bind to other apps except for themselves and the following system components.
  - `com.android.bluetooth`
  - `com.android.providers.contacts`
  - `com.android.providers.media`
  - `com.android.providers.telephony`
  - `com.android.systemui`
  - `com.google.android.providers.media.module`

- All common capabilities in the ASI APK MUST be integrated into the Android framework by overriding the following overlayable resources.
  - `config_defaultOnDeviceSpeechRecognitionService`
  - `config_defaultAugmentedAutofillService`
  - `config_defaultTextClassifierPackage`
  - `config_defaultContentCaptureService`
- Under the top-level Privacy menu in the Settings app, there MUST be an **Android System Intelligence** item, which MUST start a setting activity in the ASI APK.
- The data from the ASI app and the Private Compute Services app MUST NOT be backed up outside of the standard Android platform backup mechanism.

[GMS-7.21-006] DEVICEs preloading the Private Features APK MUST meet the following integration requirements:

- A system feature string `com.google.android.feature.ASI` MUST be declared.
- A system feature string `com.google.android.feature.ASI_MINIMAL` MUST NOT be declared.
- A system feature string `com.google.android.feature.DPS` (legacy ASI feature) MUST NOT be declared.
- The Private Features APK MUST be allowlisted to use the following additional privileged permissions.
  - `android.permission.MANAGE_APP_PREDICTIONS`
  - `android.permission.CAPTURE_MEDIA_OUTPUT`
  - `android.permission.MODIFY_AUDIO_ROUTING`
  - `android.permission.PACKAGE_USAGE_STATS`
  - `android.permission.START_ACTIVITIES_FROM_BACKGROUND`
  - `android.permission.UPDATE_DEVICE_STATS`
  - `android.permission.SUBSTITUTE_NOTIFICATION_APP_NAME`
- The Private Features APK must be configured as the system role holder of the following additional roles.
  - `SYSTEM_UI_INTELLIGENCE` (`config_systemUiIntelligence`)

    - `SYSTEM_VISUAL_INTELLIGENCE` (`config_systemVisualIntelligence`)
- All capabilities in Private Features APK MUST be integrated into the Android framework by overriding the following overlayable resources.
    - `config_defaultAppPredictionService`
    - `config_defaultSystemCaptionsService` (starting Android T)
    - `config_defaultSystemCaptionsManagerService` (starting Android T)
    - `config_defaultAttentionService`
    - `config_adaptive_sleep_available`
- Each ASI feature MUST have at least one associated entry in Settings.
- Launcher settings MUST include a link to ASI App Suggestion settings.
- Display settings MUST include a link to ASI Screen Attention settings.
- The presentation of ASI feature controls in Settings UI must not be wrapped in other branding.
- The App Suggestions feature SHOULD be turned ON by default out of box.
- The Screen Attention feature MUST be turned OFF by default out of box.

Integration guide

See Android System Intelligence (/gms/building/apps/asi) for the integration guide.

7.22 Android Messages

Android Messages (`com.google.android.apps.messaging`) provide Android DEVICEs with SMS, MMS, and RCS functionality.

[GMS-7.22-001] DEVICEs MAY preload the Android Messages app.

[GMS-7.22-002] The DEVICEs preloading the Android Messages app and:

- **[GMS-7.22-002.001]** Launching with Android 12 or higher, the Android Messages app:
    - MUST be preloaded as a privileged app in the system image, by placing it in the folders like `/product/priv-app`.

- MUST allowlist (https://source.android.com/devices/tech/config/perms-allowlist) the following privileged permissions:

```
*   `android.permission.CONNECTIVITY_USE_RESTRICTED_NETWORKS`
*   `android.permission.MODIFY_PHONE_STATE`
*   `android.permission.READ_PRIVILEGED_PHONE_STATE`
*   `android.permission.READ_PRECISE_PHONE_STATE`
*   `android.permission.START_FOREGROUND_SERVICES_FROM_BACKGROUND`
```

- **[GMS-7.22-002.002]** Running Android 10 or Android 11, the Android Messages app:
  - MUST be preloaded in the system image in the folder `/product/app`.
- **[GMS-7.22-002.003]** Running Android 9 or lower, the Android Messages app:
  - MUST be preloaded in the system image in the folder `/system/app`.

[GMS-7.22-003] It is STRONGLY RECOMMENDED to set up the Android Message app as the default SMS and messaging app.

[GMS-7.22-004] It is STRONGLY RECOMMENDED to add the Android Messages app icon to the app dock at the bottom of the default home screen.

Integration guide

See Android Messages and Carrier Services (/gms/building/apps/communication/android-messages) for the integration guide.

End new requirements

8 Best User Experience

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)

- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Release	Requirements	Links to requirements
March 2022	8.22	8.22 Reporting the location in the SIP header during a user-initiated emergency call (#sip-location)
March 2022	8.32	8.32 Dynamic color tonal palettes (created Nov 15, 2021/updated Jan 26, 2022) (#dynamic-color)

8.1 Setup experience

The GMS Setup Wizard app provides a reference implementation for the best setup experience for DEVICEs. It's STRONGLY RECOMMENDED to use the GMS Setup Wizard app as is. However, partners MAY modify or customize the setup experience as long as it meets the requirements below. See Setup Wizard Overview (/gms/building/integrating/suw) for details about the setup flow requirements and the available customizations.

- **[GMS-8.1-001]** On DEVICEs with Wi-Fi, users MUST be able to set up the Wi-Fi connection before setting up any other DEVICE capabilities that require a network connection.
- **[GMS-8.1-002]** The Google Account MUST be the first account set up to be presented to users after network connection. However, carrier account login MAY appear beforehand if the DEVICE features an eSIM AND its provisioning requires logging in to a carrier account for setup. If it appears before the Google Account setup, the eSIM provisioning implementation in the Setup Wizard MUST feature only the screens to establish network connectivity.
- **[GMS-8.1-003]** Account setup (Google and others) presented to users during the DEVICE setup MUST offer users an option to skip the setup.
- **[GMS-8.1-004]** End users MUST be presented with the Google Services page, which partners can't modify.

- **[GMS-8.1-005]** DEVICEs MUST use the Wizard Manager to customize the setup experience.
- **[GMS-8.1-006]** DEVICEs MUST comply with Android Enterprise setup experience requirements.

8.2 Setup Wizard theming

[GMS-8.2-001] DEVICEs launching with Android 10 or higher MUST implement Setup Wizard screens using the Setup Wizard compatibility library. See SetupDesign and the SetupCompat Libraries Overview (/gms/building/integrating/suw/design-compat) for implementation details.

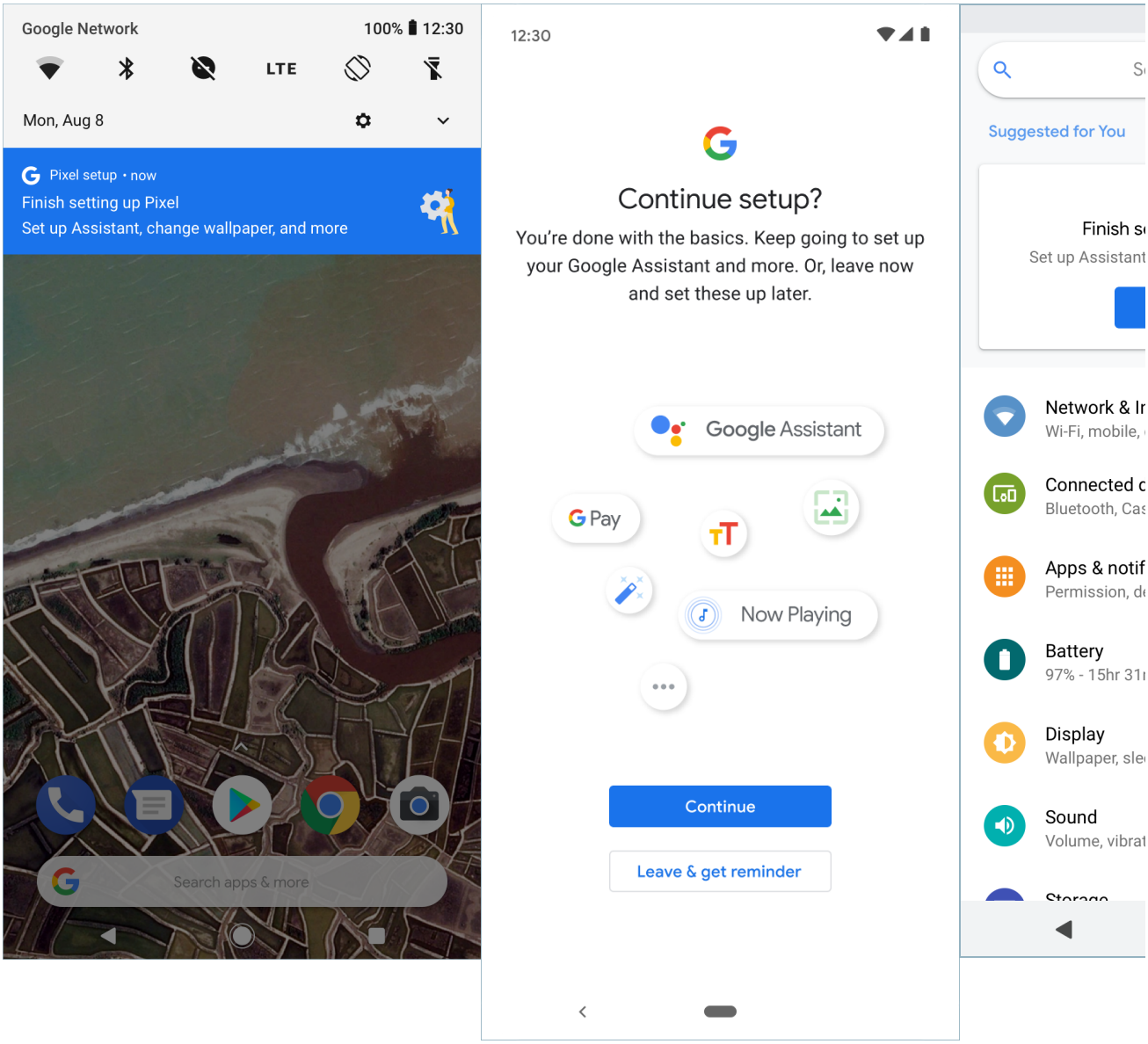
8.3 Deferred Setup

Background

The GMS Setup Wizard app supports *Deferred Setup* using the Deferred Setup Wizard (/gms/building/integrating/suw/deferred), which is started from a notification or a suggestion in Settings. Deferred Setup lets users revisit the setup screens they skipped during the out-of-box Setup Wizard. DEVICEs launching with Android 10 or higher are required to implement Deferred Setup following the technical requirements below.

Deferred Setup is **STRONGLY RECOMMENDED** for Android Go (https://docs.partner.android.com/gms/policies/domains/go) DEVICEs.

For more information, refer to Customizing Setup Wizard (/gms/building/integrating/suw/customizing-suw).

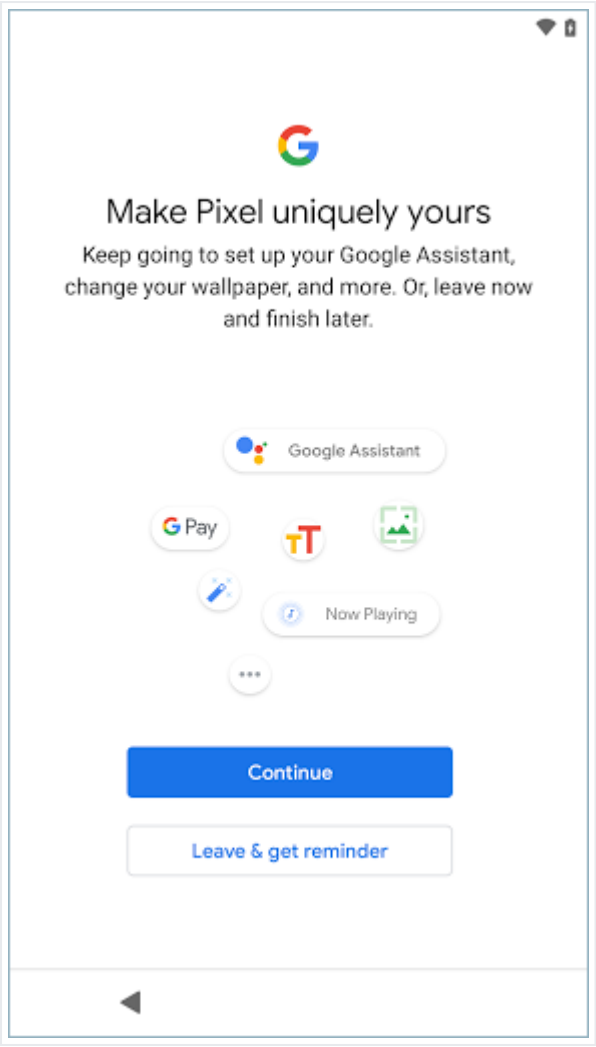


Triggering

[GMS-8.3-001] Deferred Setup MUST trigger if any of the following conditions is met.

- The user skips all connectivity setup screens (SIM and Wi-Fi).
- The user skips the Google Account sign-in screen.
- Setup Wizard was customized to show a decision point screen (/gms/building/integrating/suw/decision-point), and the user skips that screen to exit from the out-of-box Setup Wizard.

Note: Implementing the decision point screen is optional.



Partners MAY optionally configure Deferred Setup to trigger if users skip other screens, such as the secure lock screen setup, OEM/carrier account setup, or OEM/carrier service setup.

Notification

If Deferred Setup is triggered, a notification MUST be presented following these technical requirements. Partners MAY optionally enable the Deferred Setup Settings suggestion.

- **[GMS-8.3-002]** If users skip any of the three cases in Triggering (#triggering), the Deferred Setup notification MUST be nondismissible.
- **[GMS-8.3-003]** If the Deferred Setup notification is posted as nondismissible, the user MUST complete Deferred Setup by completing or skipping each screen to remove the notification.

Deferred Setup experience

[GMS-8.3-004] All Google-provided screens that are skipped in the out-of-box Setup Wizard MUST be started again in the Deferred Setup Wizard flow without modifications.

Note: Deferred Setup automatically skips some Google-provided screens if the user already completed those setup actions elsewhere. For example, if the user signed in with a Google Account from the Play Store app before starting Deferred Setup, the Google Account setup screen doesn't appear again.

Decision point in the out-of-box setup experience

- **[GMS-8.3-005]** Partners MAY add a decision point screen (/gms/building/integrating/suw/decision-point) that allows users to push all following setup screens to Deferred Setup.
- **[GMS-8.3-006]** The out-of-box setup flow preceding the decision point screen MUST still be compliant with other GMS requirements.
- **[GMS-8.3-007]** At the decision point screen (/gms/building/integrating/suw/decision-point), users MUST explicitly make a decision to skip or continue.

8.4 System navigation experience

Reference navigation models

Navigation model	Behavior
(A) Classic three-button navigation	Physically separated buttons (on-screen or hardware) for HOME, BACK, and RECENTS functions
(B) Android 9 two-button navigation	Physically separated on-screen buttons for HOME and BACK Swipe-up gesture for RECENTS (Overview)
(C) Gesture navigation	HOME gesture: Swipe-up BACK gesture: Swipe-in from left and right edge RECENTS (Overview) gesture: Swipe-up and hold

Note: Refer to System Navigation Experience (/gms/building/settings/security/system-navigation-experience) for implementation details; all edge descriptions are relative to the current orientation of the DEVICE.

Out-of-box default navigation model (new DEVICEs)

The default out-of-box (OOB) navigation MUST adhere to the requirements in the table below.

		Default OOB navigation	Remarks
[GMS-8.4-001]	DEVICEs launching with Android 8.x or lower	(A)	
[GMS-8.4-002]	DEVICEs launching with Android 9	(A) or (B) (A) is STRONGLY RECOMMENDED	Partners MAY provide (C) as a user-selectable option, but they would need to implement it by themselves.
[GMS-8.4-003]	DEVICEs launching with Android 10 or higher	(A) or (C)	If (C) is OOB default: (A) MUST be additionally supported as navigation option and at the same level as (C) in settings. (B) MUST NOT be provided as a user-selectable option.

Navigation model

[GMS-8.4-004] DEVICEs launching or upgrading to Android 10 or higher are STRONGLY RECOMMENDED to keep the current navigation UX upon OTA update. However, option (C) can be added as an option for the user.

Opt-in navigation

[GMS-8.4-005] Partners MAY ship opt-in navigation models that have a different UX from the reference navigation models. However, the opt-in navigation models MUST NOT be advertised by the Setup Wizard or any other method, such as notifications or on-screen popups.

[GMS-8.4-006] DEVICEs launching with Android 10 or higher MUST have a system navigation settings menu if multiple navigation models are supported. If both (A) and (C) are supported, the setting to switch between two models MUST be placed at the top level of the system navigation settings, and any alternative opt-in navigation models MUST be placed only in the same settings category, but one level deeper, such as under **Advanced** or similar.

[GMS-8.4-007] DEVICEs launching with Android 10 or higher MAY implement an opt-in navigation model with gestures. However, its behavior MUST comply with the Android CDD, and such DEVICEs MUST implement model (C) as well, and their navigation settings MUST satisfy [GMS-8.4-004].

User training

[GMS-8.4-008] DEVICEs launching with Android 10 or higher that support the reference gesture navigation model as out-of-box default are STRONGLY RECOMMENDED to provide training or demo screens for users.

Third-party launchers

[GMS-8.4-009] If a user sets a third-party launcher as the default launcher, the system is STRONGLY RECOMMENDED to not switch the user to an alternate navigation mode ((A), (B), (C), or opt-in navigation modes).

8.5 Default Assist app

[GMS-8.5-001.001] On DEVICEs running Android 9 or lower:

- Long-pressing the **home** button (hardware key or on-screen software button) MUST always bring up the user-selected default Assist app.
- As the default Assist app, users may select any app that supports either the `VoiceInteractionService` (https://developer.android.com/reference/android/service/voice/VoiceInteractionService) class (as implemented by AOSP) or the Assist action handler. However, the Google app MUST be the out-of-box default Assist app.

[GMS-8.5-001.002] On DEVICEs running Android 10 or higher:

- If the reference gesture navigation (as defined in System navigation experience (#system-nav-experience)) is supported, the swipe-up/swipe-in gesture from the bottom-left/bottom-right corner MUST bring up the user-selected default Assist app. This gesture MUST be implemented regardless of whether the reference gesture navigation is the out-of-box default or provided as a user-selectable option.
- The technical specifications for Assistant gesture navigation MUST be compliant with the requirements detailed in System Navigation Experience (/gms/building/settings/security/system-navigation-experience).

[GMS-8.5-002] If a nonreference navigation model is provided as a user-selectable option, the nonreference navigation model is subject to Google approval for how the default Assist app invocation should work.

8.6 Backup agent compatibility

The Android platform can back up user data such as settings, installed apps, and app data to a user's private cloud storage that belongs to their Google Account.

This lets users promptly set up a new Android phone or tablet by restoring their backup data in the cloud. The GMS bundle includes all binaries, framework overlays, and config files for partners to implement such backup and restore experiences. DEVICEs MUST support this feature according to the requirements below.

[GMS-8.6-001] Backup agents defined in AOSP SHOULD NOT be modified in ways that prevent interoperation with other DEVICEs. For example, the package name of the AOSP package having its backup agent MUST NOT be changed. In particular, GTS tests ensure correct backup and restore of the following packages:

- `android`
- `com.android.calllogbackup`
- `com.android.wallpaperbackup`
- `com.android.providers.settings`
- `com.android.providers.telephony`
- `com.android.providers.blockednumber`
- `com.android.providers.userdictionary`

8.7 Secure texture video playback

[GMS-8.7-001] Secure texture video playback is optional for Widevine DRM L1 DEVICEs.

8.8 Autofill provider

[GMS-8.8-001] DEVICEs running Android 8.0 or higher are RECOMMENDED to support and declare the `android.software.autofill` feature. The Google Play services app in the GMS bundle provides an implementation of the Autofill provider.

8.9 App Crash dialog

[GMS-8.9-001] Android 9 changed the behavior of the App Crash dialog. It no longer appears for a one-time crash for both foreground and background apps. However, if an app crashes two or more times successively, the App Crash dialog still appears, giving the user the option to view app info or force close the app. DEVICEs running Android 9 or higher MUST NOT modify this behavior.

8.10 Rich Communications Services (RCS)

Google believes that interoperable IP messaging based on the open Rich Communications Services (RCS) specification is a viable direction to improve the messaging experience beyond SMS/MMS for Android users.

[GMS-8.10-001] DEVICEs with `FEATURE_TELEPHONY` (https://developer.android.com/reference/android/content/pm/PackageManager#FEATURE_TELEPHONY) support that seeks approval on or after:

- **February 1, 2019**:
    - MUST include a default telephony messaging app. A default telephony messaging app is the default handler for the `SMS_DELIVER_ACTION` (https://developer.android.com/reference/android/provider/Telephony.Sms.Intents#SMS_DELIVER_ACTION) intent. The default telephony messaging app:

- MUST support the RCS Universal Profile 1.0 and the following associated services:
  - Device provisioning
  - Capability discovery
  - 1-to-1 messaging
  - Group chat
  - File transfer
- MUST NOT be affected by Android's battery optimization (Doze mode), so that the app can receive messages in a timely fashion.

Note: The app SHOULD implement the most up-to-date RCS Universal Profile version.

8.11 Carrier ID

[GMS-8.11-001.001] Android 9 introduced a carrier ID database (https://source.android.com/devices/tech/config/carrierid) published in AOSP as `packages/providers/TelephonyProvider/assets/carrier_list.textpb`. DEVICEs running Android 9 or higher MUST NOT alter this implementation or the content of the database.

Google updates the database regularly with the latest version using the GMS ConfigUpdater app, which downloads new carrier ID data from Google servers to the internal storage folder at `/data/misc/carrierid/`. DEVICE implementations MUST handle `android.os.action.UPDATE_CARRIER_ID_DB` and persist the downloaded data to the carrier ID database if its version is newer than the installed version of the carrier ID database. After the carrier ID database is updated through this mechanism, the behavior of public APIs MUST follow the updated database. PRODUCTs launching with Android 9 or higher MUST use the latest version of the carrier ID database published in AOSP.

Partners can suggest additions to the carrier ID data by filling out the external carrier identification information form (https://docs.google.com/forms/d/1KjwTaExKRjkE9tbR9yavBrGzwvuz1dNku2Ae_7GrdUQ/viewform?edit_requested=true)

.

See Carrier Identification (https://source.android.com/devices/tech/config/carrierid) for more information about implementation.

[GMS-8.11-001.002] Android 10 or higher support using a carrier ID (returned by `getSimCarrierId`) to load carrier configurations from `CarrierService` (https://developer.android.com/reference/android/service/carrier/CarrierService). Android supports out-of-band carrier ID database updates, as described in this section. It's possible a mobile virtual network operator (MVNO) was previously unrecognized, so it inherited an ID from its mobile network operator (MNO) by default. The new carrier list update may include a new carrier ID for the missing MVNO. Implementing `CarrierService` is STRONGLY RECOMMENDED to support MNO carrier ID fallback by using the `getCarrierIdFromSimMccMnc` method in the case that config data from `CarrierService` isn't updated and included in the new MVNO ID.

8.12 USB Type-C compatibility

[GMS-8.12-001] DEVICEs launching from 2019 onward with a USB Type-C port MUST ensure full interoperability with chargers that are compliant with the USB specifications and have a USB Type-C plug.

[GMS-8.12-002] DEVICEs launching from October 2018 onward with a USB Type-C port:

- When bundling an audio headset with USB Type-C plug in the retail product packaging, it MUST be a digital headset because the USB standards forbid analog audio headsets to use the USB Type-C plug, and it potentially creates confusion for users who might plug such a headset into other Android DEVICEs approved for GMS.
- When bundling a USB Type-C to a 3.5 mm analog audio jack conversion dongle in the retail product packaging, it SHOULD be a USB Type-C digital-to-analog audio dongle for interoperability across DEVICEs.

8.13 Low-latency audio

[GMS-8.13-001] If they meet certain hardware requirements, DEVICEs launching with Android 9 or higher are STRONGLY RECOMMENDED to support low-latency, MIDI, and pro audio features as below:

- DEVICEs with 2 GB or more RAM: `FEATURE_AUDIO_LOW_LATENCY` (https://developer.android.com/reference/android/content/pm/PackageManager#FEATURE_AUDIO_LOW_LATENCY)

and `FEATURE_MIDI` (https://developer.android.com/reference/android/content/pm/PackageManager#FEATURE_MIDI)

- DEVICEs with 4 GB or more RAM: `FEATURE_AUDIO_PRO` (https://developer.android.com/reference/android/content/pm/PackageManager#FEATURE_AUDIO_PRO)

8.14 Acoustic echo canceler (AEC)

[GMS-8.14-001] DEVICEs launching with Android 10 or higher, which provide an acoustic echo canceler (AEC) inserted in the capture audio path when `AudioSource.VOICE_COMMUNICATION` is selected, MUST:

- Declare it with the AEC API method `AcousticEchoCanceler.isAvailable()`.
- Allow this audio effect to be controllable with the `android.media.audiofx.AcousticEchoCanceler` API.
- Uniquely identify each AEC technology implementation with the `AudioEffect.Descriptor.uuid` field.

8.15 ADB factory reset and Test Harness mode

[GMS-8.15-001] DEVICEs running Android 10 or higher MUST support ADB Test Harness mode as implemented in AOSP.

8.16 Power management

Android includes enhanced power management features (http://source.android.com/devices/tech/power/mgmt), also known as App Standby and Doze mode. The GMS bundle includes all necessary framework overlays and configuration files to enable these features. For more information, refer to Platform Power Management: Doze (https://source.android.com/devices/tech/power/platform_mgmt#doze) and App standby (https://source.android.com/devices/tech/power/app_mgmt#app-standby).

[GMS-8.16-001] Android DEVICEs with a significant motion detection sensor (https://source.android.com/devices/sensors/sensor-types#significant_motion) MUST enable Doze mode and MUST NOT deviate from AOSP implementation.

[GMS-8.16-002] To ensure that Google services (for example, Cloud Messaging) behave correctly, the GMS bundle provides a system configuration file to prevent some GMS apps from falling into battery saving mode. DEVICEs MUST NOT modify this configuration unless approved by Google.

App standby buckets

[GMS-8.16-003] Android 9 introduced app standby buckets, which let the platform throttle the apps based on their usage. If this feature is enabled, the throttling parameters MUST be updatable by Google to ensure consistency for app developers. Partners MAY optionally preload the Google Device Health Services app to update the app buckets.

App restrictions

[GMS-8.16-004] Partners MUST NOT apply restrictions on apps outside of:

- Low battery mode: `PowerManager.isPowerSaveMode()` (https://developer.android.com/reference/android/os/PowerManager#isPowerSaveMode())
- App restrictions: `ActivityManager.isBackgroundRestricted()` (https://developer.android.com/reference/android/app/ActivityManager#isBackgroundRestricted())

8.17 Data Saver

The Data Saver feature allows users to control which apps can access data in the background and which can access data only while in the foreground. This ensures desired the background data exchange when Data Saver is on per user control.

[GMS-8.17-001] DEVICEs MUST support the Data Saver feature.

[GMS-8.17-002] However, Google Play services MUST be allowlisted from the Data Saver mode to ensure that its important features (for example, Cloud Messaging) work in the background. The GMS bundle releases provide all necessary files to build devices that comply with this requirement. For the detailed requirements, refer to Data Saver mode (https://source.android.com/devices/tech/connect/data-saver).

8.18 Default print service

[GMS-8.18-001] DEVICEs running Android 10 or higher MUST preinstall at least one print service. Partners can use the *default print service* in AOSP or some 3P services. If the default print service is installed, then it SHOULD be enabled.

8.19 Emergency number database

Starting in Android 11, GMS Core supports the update of the emergency number database regularly with the latest version, which downloads the new emergency number database from Google servers to the internal storage folder in the device.

[GMS-8.19-001] Device implementations MUST handle `android.os.action.UPDATE_EMERGENCY_NUMBER_DB` and persist the downloaded data to the emergency number database if its version is newer than the installed version of the emergency number database. When the emergency number database is updated through this mechanism, the behavior of public APIs MUST follow the updated database. PRODUCTs launching with Android 11 or higher MUST use the latest version of the emergency number database.

Partners can suggest additions to the emergency number data by filling out the External Emergency Number Database Request form (https://docs.google.com/forms/d/1TPXvMMTycDOOsr0xMZQtabwIJfQnqSYDsnxY1ZyPVKs).

8.20 Incremental APK support

[GMS-8.20-001] DEVICEs launching with Android 11 or higher MUST:

- Have incremental file system (IncFS) kernel driver module loaded at boot.
- Support verifying APK files using the APK Signature Scheme v4.

[GMS-8.20-002] DEVICEs launching with Android 11, and with `FEATURE_INCREMENTAL_DELIVERY` support, MUST add `android.intent.extra.DATA_LOADER_TYPE` and set its value to one of the following (`none`, `streaming`, or `incremental`) depending upon the data loader used for package installation to the `ACTION_PACKAGE_ADDED` broadcast.

8.21 Emergency dialer access

[GMS-8.21-001] DEVICEs launching with Android 11 or higher and DEVICEs upgrading from Android 11 to Android 12 or higher and having `FEATURE_TELEPHONY` MUST include a button on the lock screen that lets users make emergency calls in the default dialer app. This button MUST:

- Include the word **Emergency**.
- Be placed under the number pad but above the unlock mechanism. Note: *Number pad* should clearly note that this is for unlocking your device.
- Be located in the lower 40% of the screen. It's **STRONGLY RECOMMENDED** to place the emergency button at the bottom center of the screen. The keyboard MAY push the button up.

8.22 Reporting the location in the SIP header during a use initiated emergency call

[GMS-8.22-001] For DEVICEs launching Android 11 or higher, if they support `android.hardware.telephony`, and a SIP stack that lets end users make SIP calls through the licensed network operator, it's a MUST that:

- When a SIP session is initiated during a user-initiated emergency session (voice/video/RTT calls), the device implementation MUST populate the available

Location Estimates (#loc-estimates) in the `SIP_INVITE` (MAY populate in `SIP_REGISTER`) message with the best available Location Estimates (#loc-estimates) (Fused Location Provider (FLP) if the provider is available, if not available, use the Location Manager).

- Only during a user-initiated emergency session, if the users location toggle is set to off, the OEM MUST implement the Emergency Location Bypass API to be able to get the users FLP or Location Manager Location during this emergency session. DEVICEs MUST NOT populate any Location Estimates (#loc-estimates) outside of a user-initiated emergency session in the SIP message unless the user has provided consent through location settings and app-specific permissions.

8.23 Device Controls

Device Controls is a new AOSP feature for 3P smart home or IOT device control apps to show up on a new UX surface. This feature provides partners with an option to configure one or more apps of their choice to be prefavored on the surface. Partners are required to add Google Home as one of the prefavored apps as detailed in the requirements.

DEVICEs running Android 11 or higher:

- **[GMS-8.23-001]** MUST implement the Device Controls user affordance as defined in CDD section [3.8.16] (https://source.android.com/compatibility/android-cdd#3816_device_controls).
- **[GMS-8.23-002]** MUST configure the Google Home (https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app) app to be one of the prefavored apps for a Device Controls user affordance, as explained in Device Controls (/gms/building/settings/overview/device-controls). When apps that are configured to be prefavored are installed on the device, they're added to the list of switchable providers in Device Controls.
- **[GMS-8.23-003]** MAY have another app configured as the prefavored app for Device Controls and append the name of the package in the configuration setting. The Google Home app doesn't have to be the first item in the prefavored apps list.

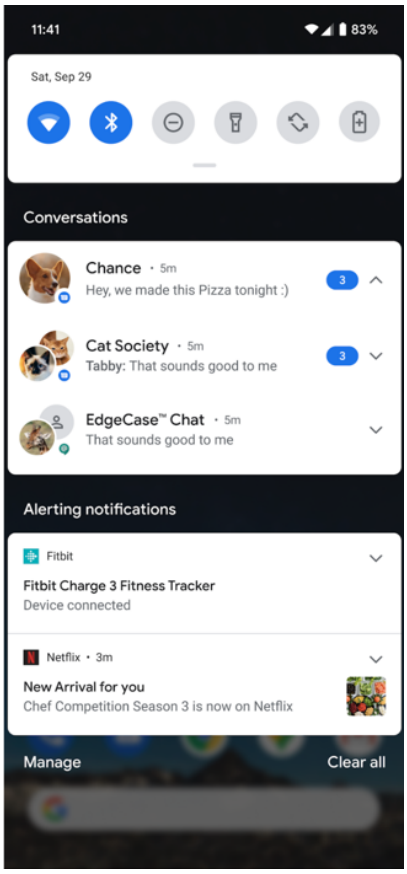
Note: The Google Home app package itself is NOT REQUIRED to be pre-installed.

This requirement is NOT applicable to Android Go devices.

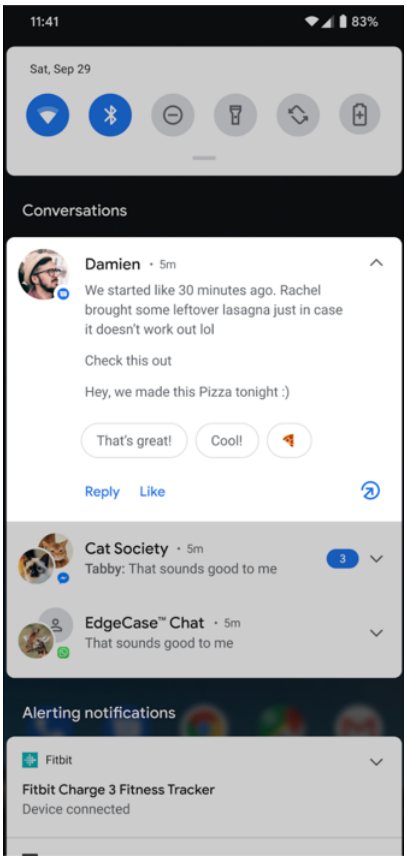
8.24 Conversations

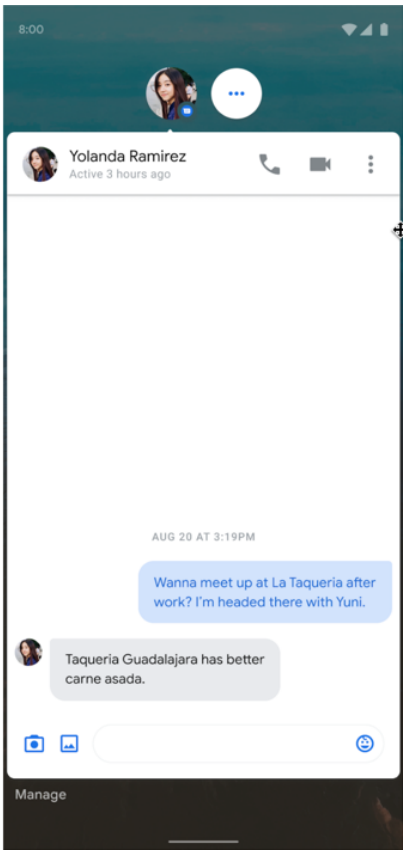
Android 11 introduced support for conversation notifications, which are notifications that use `NotificationManager.MessageStyle` and provide a published People Shortcut ID.

[GMS-8.24-001] DEVICEs launching or upgrading to Android 11 or higher MUST group and display conversation notifications ahead of nonconversation notifications with the exception of ongoing foreground service notifications and `importance:high` notifications.



[GMS-8.24-002] DEVICEs launching or upgrading to Android 11 or higher MUST provide the user access to display a conversation as a bubble directly from the notification associated with the conversation if the app generating the notification supports bubbles.





[GMS-8.24-003] For DEVICEs launching or upgrading to Android 11, when bubbles are open, it's **STRONGLY RECOMMENDED** to present the recent conversations UI (**+** button) for users to be able to tap into and create new bubbles.

[GMS-8.24-004] For DEVICEs running Android 12 or higher, when bubbles are open, they **MUST** present the recent conversations UI with any obvious affordance (STRONGLY RECOMMENDED to use the **+** button) for users to be able to tap into and create new bubbles.

8.25 Ambient display

Android 11 introduced an API method to suppress a DEVICE's ambient display. Ambient display is defined as anything visible on the display when the DEVICE isn't interactive (that is, `PowerManager.isInteractive()` (https://developer.android.com/reference/android/os/PowerManager.html#isInteractive()) returns `false`).

- **[GMS-8.25-001]** `PowerManager.isAmbientDisplayAvailable()` MUST return `true` if and only if the DEVICE supports ambient display.

- **[GMS-8.25-002]** If the DEVICE supports ambient display and at least one app has called `PowerManager.suppressAmbientDisplay(token, true)` with at least one `token` instance, then the DEVICE MUST temporarily disable ambient display. The ambient display MUST remain disabled until all apps that previously called `PowerManager.suppressAmbientDisplay(token, true)` call `PowerManager.suppressAmbientDisplay(token, false)` for the relevant `token` instances or until the DEVICE reboots.

8.26 Event logging in System UI and Launcher

[GMS-8.26-001] The System UI and the preloaded default launchers MUST implement the same event logging mechanism provided in the AOSP. For more information, refer to Implementing Launcher Workspace Logging (https://drive.google.com/file/d/0B7itl8AIz03UV0NhNGJiOFVVRGM/preview).

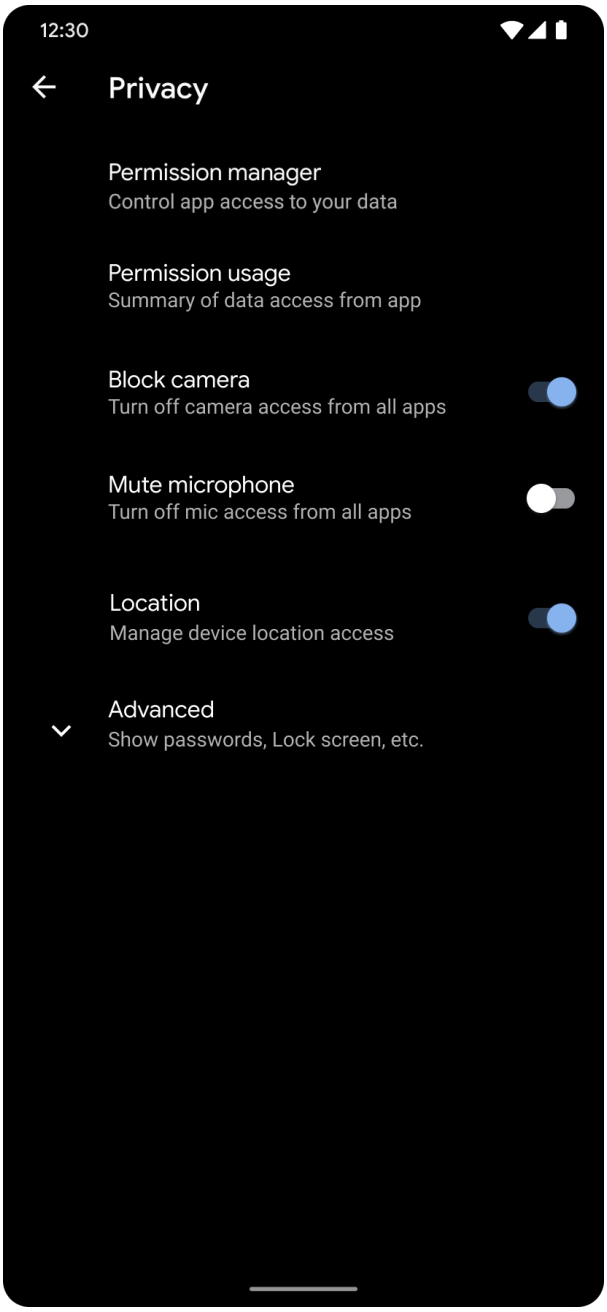
8.27 Updatable system font

[GMS-8.27-001] DEVICEs launching with Android 12 or higher MUST allow the runtime update of `NotoColorEmoji.ttf`.

8.28 Microphone and camera toggles

[GMS-8.28-001] On DEVICEs running Android 12 ~~or higher~~, if `android.hardware.SensorPrivacyManager.supportsSensorToggle(int)` = `TRUE` then the DEVICEs MUST meet the following requirements:

- The microphone and camera toggles MUST be placed in **Settings** > **Privacy**.
- The microphone and camera toggles in the Privacy Settings page MUST clearly communicate the option to block the camera and mute the microphone.
- When the microphone or camera toggle is OFF, the DEVICE MUST NOT provide the real (or actual) camera and audio data to other systems on the DEVICE outside of the Android framework.



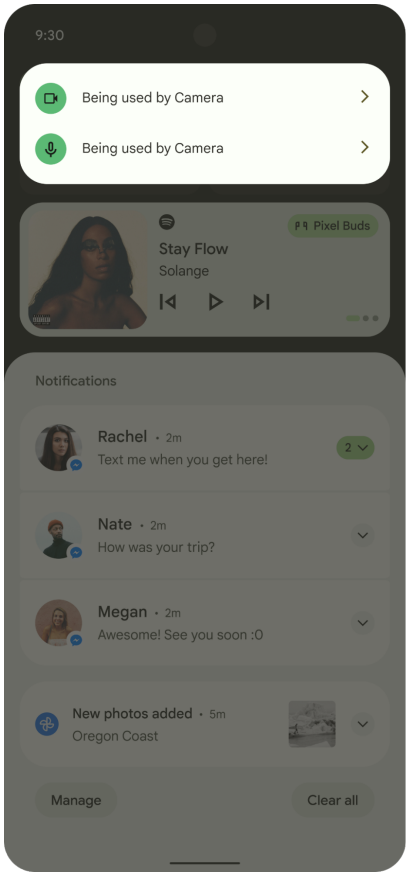
8.29 Microphone and camera indicators

[GMS-8.29-001] On DEVICEs running on Android 12 or higher, microphone and camera indicators MUST meet the following requirements:

- The microphone and camera indicators MUST appear in the status bar as the highest priority status icon (for example, rightmost position in the top-right corner).
- The microphone and camera indicators MUST be constantly visible in the same position, even on a full screen overlay (immersive) UI and position indicated to the apps by `WindowInsets.getPrivacyIndicatorBounds`, which reserves the boundary for

the indiators. For example, launching a camera app shouldn't block the user's view of the indicators.

- Both microphone and camera indicators MUST be in green or a variation of green.
- Clicking the microphone and camera indicators MUST render an app attribution affordance notification, which:
  - Displays the name of the app that is currently using microphone or camera if the access is current or the name of the app that recently used microphone or camera if the access occurred in the last 15 seconds.
  - Takes users to the app permission page in **Settings** using the `android.intent.action.MANAGE_APP_PERMISSIONS` intent, if the affordance is clicked.



8.30 Telephony

[GMS-8.30-001] DEVICEs launching with Android 12 that support IWLAN technology, MUST support the AP-assisted mode for IWLAN by setting the

`ro.telephony.iwlan_operation_mode` system property to `IWLAN_OPERATION_MODE_AP_ASSISTED` or to `IWLAN_OPERATION_MODE_DEFAULT`. Alternatively, partners can leave the system property `ro.telephony.iwlan_operation_mode` undefined.

8.31 Ultra wideband (UWB)

[GMS-8.31-001] If DEVICEs launching with Android 12 include UWB hardware, they MUST enable the `android.hardware.uwb` feature.

Start new requirements for 12

8.32 Dynamic color tonal palettes

Android 12 introduces the following dynamic color tonal palettes each comprising a set of 13 colors with defined various luminance values as described in `R.color` (https://developer.android.com/reference/android/R.color) but with undefined hue and chroma values that can be dynamically generated by the Android system at runtime.

- `system_accent1_0` (https://developer.android.com/reference/android/R.color#system_accent1_0)
- `system_accent2_0` (https://developer.android.com/reference/android/R.color#system_accent2_0)
- `system_accent3_0` (https://developer.android.com/reference/android/R.color#system_accent3_0)
- `system_neutral1_0` (https://developer.android.com/reference/android/R.color#system_neutral1_0)
- `system_neutral2_0` (https://developer.android.com/reference/android/R.color#system_neutral2_0)

Dynamic color tonal palettes requirements

DEVICEs launching with Android 12 or higher, submitted for approvals from **March 14, 2022**:

OR

PRODUCTs launching or upgrading to Android 12 or higher, submitted for approvals from **March 14, 2022**:

[GMS-8.32-001] MUST generate the dynamic color tonal palettes from a single **source color**, which

- MUST have a CAM16 (https://onlinelibrary.wiley.org/doi/10.1002/col.22131) chroma value of 5 or higher.

[GMS-8.32-002] MUST fulfill the following requirements.

- MUST generate each of the 5 tonal palette's 13 color values with the respective changes to the source color's CAM16 (https://onlinelibrary.wiley.com/doi/10.1002/col.22131) chroma and hue values.
  - `system_accent1_0` (https://developer.android.com/reference/android/R.color#system_accent1_0):
    - With the source color's hue value
    - Chroma value 40 for colors palette indices ending in one of the following OR the maximum of the source color's chroma and 48:
    - `0` (https://developer.android.com/reference/android/R.color#system_accent1_0)
    - `10` (https://developer.android.com/reference/android/R.color#system_accent1_10)
    - `50` (https://developer.android.com/reference/android/R.color#system_accent1_50)
    - `100` (https://developer.android.com/reference/android/R.color#system_accent1_100)
  - `system_accent2_0` (https://developer.android.com/reference/android/R.color#system_accent2_0): Chroma value 16, with the source color's hue value.
  - `system_accent3_0` (https://developer.android.com/reference/android/R.color#system_accent3_0): Chroma value 32, with the source color's hue value rotated by 60 degrees positive.
  - `system_neutral1_0` (https://developer.android.com/reference/android/R.color#system_neutral1_0): Chroma value 4, with the source color's hue value.

- `system_neutral2_0` (https://developer.android.com/reference/android/R.color#system_neutral2_0): Chroma value 8, with the source color's hue value.

[GMS-8.32-003] Each of the 13 tonal palette values MUST have a specific CIELAB (https://en.wikipedia.org/wiki/CIELAB_color_space#CIELAB) Perceptual Lightness (L*) value as stated in its documentation, for example, `R.color#system_accent1_10` (https://developer.android.com/reference/android/R.color#system_accent1_10) MUST have an L* value of 99.

Integration

See Using Dynamic Color (/gms/building/ui/dynamic-color) for the integration guide.

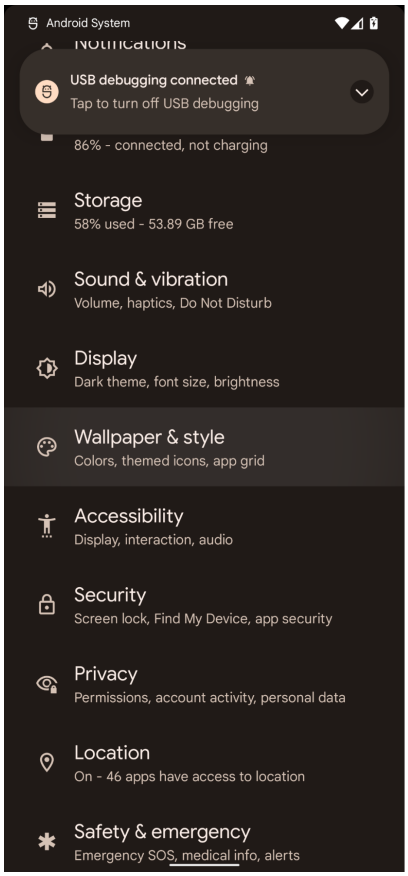
Tests

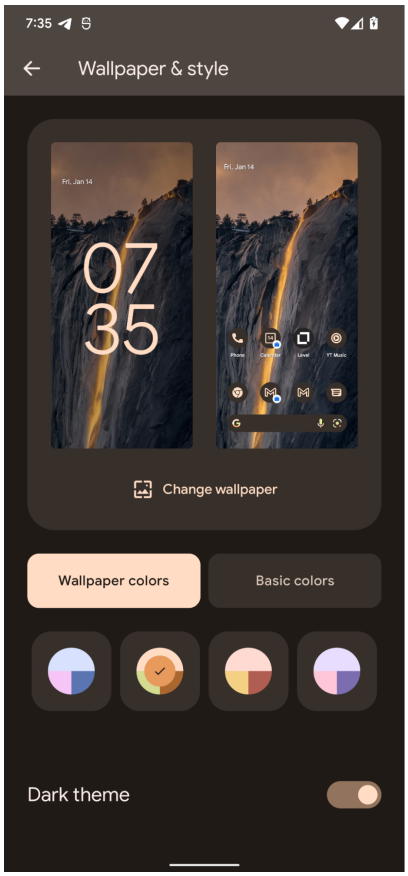
The automated test is available at `vendor/xts/gts-tests/tests/theming/src/com/google/android/theming/gts/MaterialYouPaletteTest.k`, which validates that either:

- Dynamic colors aren't implemented, and the baseline palettes weren't modified.
- Dynamic colors are implemented, and the palettes are generating valid colors.

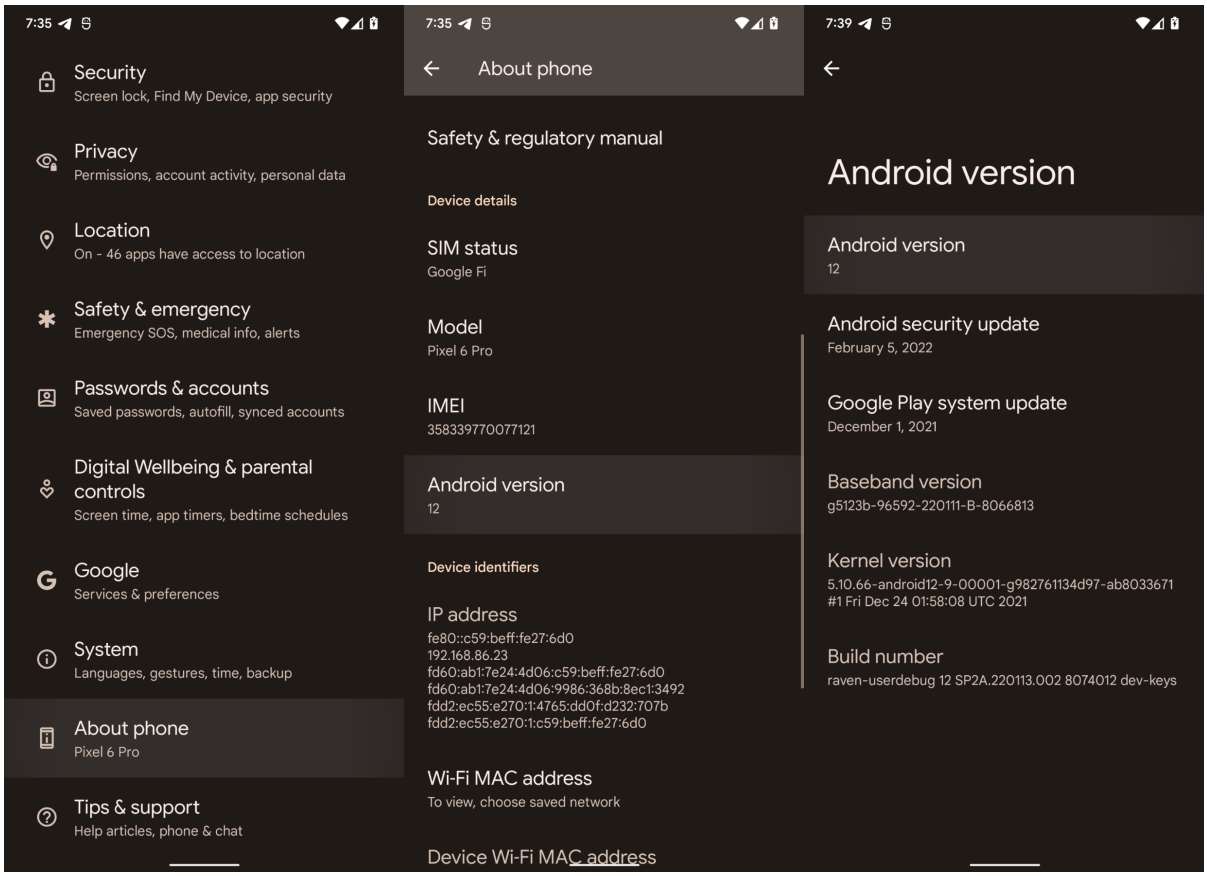
Perform the following steps to check.

1. Open the style picker and choose a theme. Each partner might have their own picker, these are screenshots from Pixel.



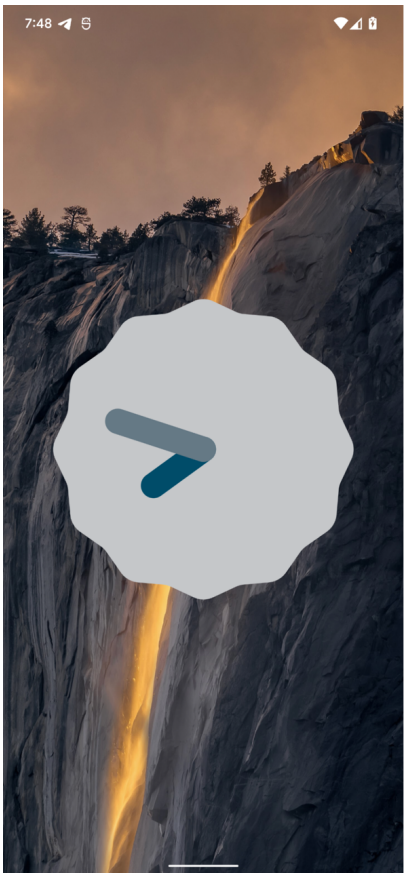


2. Open the Android 12 easter egg: **Settings > About Phone > Android Version** and tap **Android Version** until you see the Material You clock.



3. The clock must have theme colors if the OEM claims to support Material You.
    - Set the clock to midnight to see the second screen.

- Or blue + purple if they don't.



End new requirements

9 Protect Users

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Requirements	Links to Domains
9.2	Google Play Protect (#play-protect)
9.3	Security Status Overview indicator (#security-status-indicator)
9.7	DNS privacy (#dns-privacy)
9.10	Passpoint (#passpoint)
9.11	Wi-Fi MAC address randomization (#mac-address-randomization)
9.14	Domain verification asset link enforcement (#domain-verif)
9.15	Safety and Emergency Settings ; Emergency SOS (#safety-emergency)
9.16	Privacy dashboard (#privacy-dashboard)

9.17	Package Installer app (#package-installer)
9.18	Activity Recognizer (#activity-recognizer)
9.19	Deprecate insecure VPN types (#vpn-types)

9.1 Factory reset protection

DEVICEs MUST have support for factory reset protection (FRP). This feature deters theft and in some locations may be required by law.

[GMS-9.1-001] The FRP implementation MUST provide users with an OOB setup flow to turn on the protection feature. (For example, by setting up a secure lock screen) If the protection feature was turned on for a device, and it is factory reset in untrusted manner, the device MUST be unusable until its legitimate user successfully authenticates with a valid credential.

Note: Partners are RECOMMENDED to implement a recovery process that allows clearing the factory reset token, thereby enabling factory reset without user authentication. For this reason, this process must be secure and only accessible by authorized customer support personnel.

9.2 Google Play Protect

[GMS-9.2-001] On DEVICEs, Google Play store app MUST be functioning as the package verifier, which allows the Google Play Protect (https://www.android.com/play-protect/) feature to protect the users from harmful apps.

[GMS-9.2-002] If Google Play Protect flags and disables an app, the app can be reenabled in one of two ways: by Google Play Protect or by explicit user interaction.

Start new requirements for 12

~~**[GMS-9.2-003]** DEVICEs that are compliant with CDD section 9.10 Device Integrity, [C-0-3] MUST ship with Google's certificate file as the trusted fs-verity key for verifying apps continuously.~~

[GMS-9.2-003] DEVICEs that are compliant with CDD section 9.10 Device Integrity, [C-0-3] MUST ship with Google's certificate files as the trusted fs-verity keys.

End new requirements

9.3 Security settings

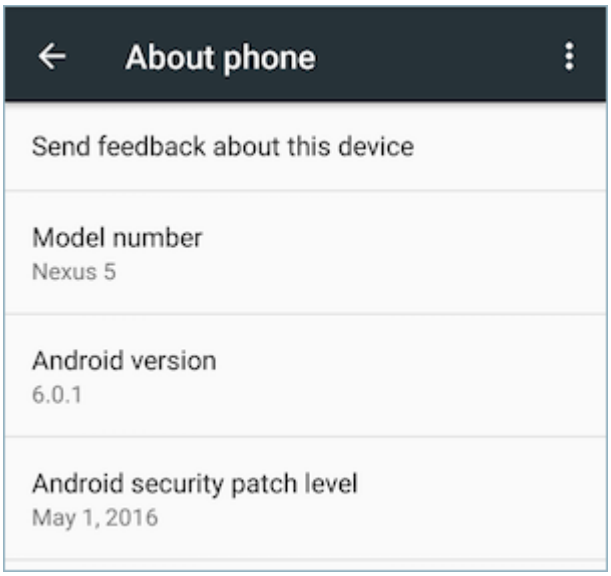
Unknown sources

[GMS-9.3-001] Android uses the **Allow app installs** setting on a per-app basis. Turning on this setting grants the `REQUEST_INSTALL_PACKAGES` (https://developer.android.com/reference/android/Manifest.permission#REQUEST_INSTALL_PACKAGES) permission that allows the granted app to provide other apps for installation. DEVICEs MUST NOT implicitly grant this permission to any preloaded or downloaded third party app.

Note: This change isn't applicable to the privileged apps having `INSTALL_PACKAGES` (https://developer.android.com/reference/android/Manifest.permission.html#INSTALL_PACKAGES) permission.

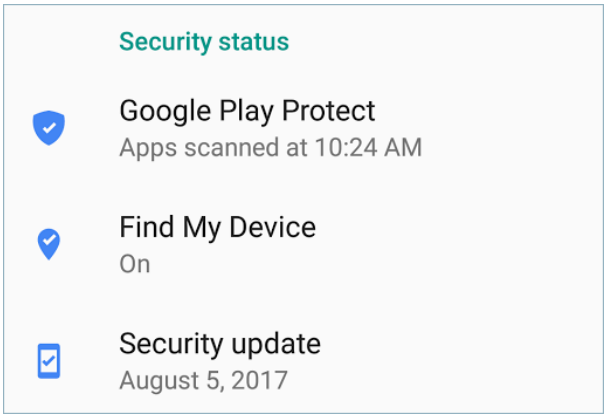
Security patch level settings

[GMS-9.3-002] DEVICEs running Android 10 or lower MUST expose the Android security patch level in the Settings UI. It MUST be displayed below the Android version item in the **About** section. The displayed text MUST match the value of the system property `ro.build.version.security_patch`, which MUST be in the format **YYYY-MM-DD** according to the CDD section 3.2.2. Build Parameters.



Security status injection into Settings

[GMS-9.3-003] The Settings app for Android 8.0 or higher DEVICEs MUST display three status items injected by the Google Play services app as shown on the screenshot below. The items MUST be grouped together under **Security status** header, MUST be visible without scrolling, and MUST be made available within single depth from the top level Settings menu. For more information, refer to Security Status in Settings (/gms/building/settings/security/security-status-settings).



Top-level security setting

[GMS-9.3-004] DEVICEs running Android 10 or higher MUST feature the **Security** section as a top-level item in the Settings app.

Note: Featuring the GPP dashboard in the Security settings already exists in the GMS Requirements.

Start new requirements for 12

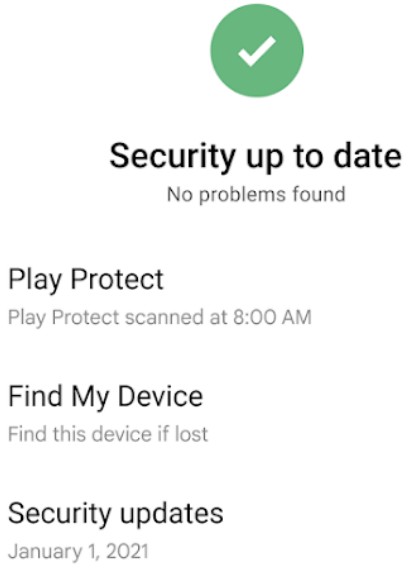
Security Status Overview indicator

[GMS-9.3-005] The Settings app for Android 12 or higher MAY display a device security status overview indicator. If implemented, the overview indicator replaces the requirement in **[GMS-9.3-003]** and MUST meet the below requirements:

- MUST include at least three security warning levels.
- MUST be visible without scrolling.
- MUST be made available within single depth from the top level **Settings** menu.

For more information, see Security Status Overview Indicator (/gms/policies/domains/security-status).

Note: This requirement is optional in Android 12.



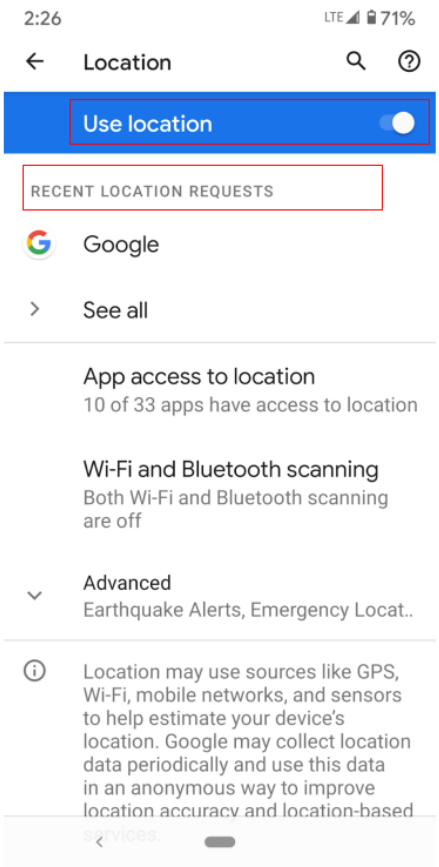
End new requirements

9.4 Location settings

[GMS-9.4-001]

The **Location** section MUST:

- Be featured as a top-level item in the **Settings** app on DEVICEs running Android 10 or higher.
- Feature the **Location Master Switch** and the **Recent Location Request** list, as implemented by AOSP.



9.5 Keystore, Keymaster 2, and key attestation

Keymaster 2 HAL extends the capabilities of hardware-backed key storage on Android DEVICEs. One of the features is key attestation, which allows Android apps and off-DEVICE entities to determine if the keys are hardware backed. If Keymaster 2 is supported, Android DEVICEs are RECOMMENDED to provision Google-provided attestation keys from the factory. The keyboxes are distributed through the Partner Approvals portal (https://partner.android.com/approvals) (see user guide (/partners/guides/approvals)). For the detailed requirements, refer to the "Keymaster2—Attestation Key Provisioning" doc in the GMS Help Center (/gms/resources/eap/android-7).

[GMS-9.5-001] PRODUCTs launching with Android 8.0 or higher MUST implement a hardware-backed keystore and provision Google-provided attestation keys from the factory.

Device ID attestation

[GMS-9.5-002] If a DEVICE implements device ID attestation, the hardware identifiers used to perform device ID attestation MUST be provisioned in the factory such that they're only accessible within the hardware keystore. There MUST be a way for users to erase these identifiers using a command line tool or similar.

Verified Boot attestation

[GMS-9.5-003] Device implementations MAY include the `verifiedBootHash` field in the key attestation certificate returned with `KeyStore.getCertificateChain()`, as defined in the Android Developers site (https://developer.android.com/training/articles/security-key-attestation#certificate_schema_rootoftrust). If the field is included, its value MUST be equal to the VBMeta digest calculated using SHA-256 as described in the `external/avb/README.md` document (https://android.googlesource.com/platform/external/avb/+/master/README.md#The-VBMeta-Digest).

[GMS-9.5-004] DEVICEs launching with Android 10 or higher MUST implement keystore requirements as defined in CDD Section 9.11, [C-1-1] to [C-1-5].

Flash lock status reporting

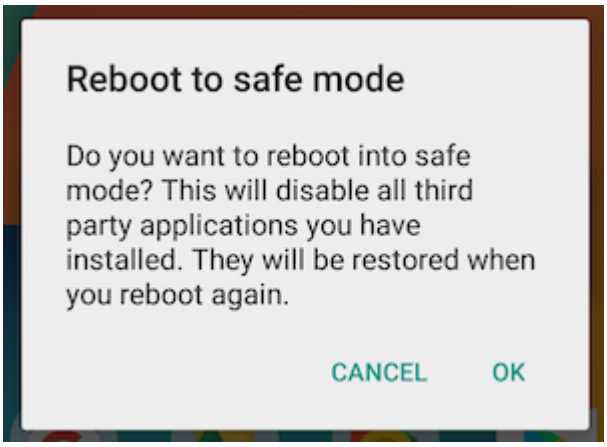
[GMS-9.5-005] Android DEVICEs MUST implement `PersistentDataBlockManager#getFlashLockState()` system API that returns the bootloader state whether it permits flashing of the system image. Its return value MUST be one of the following:

- `FLASH_LOCKED`: The bootloader doesn't allow to flash system image at all, or it's in the state that flashing of system image isn't possible.
- `FLASH_UNLOCKED`: The bootloader is in the state that allows to flash system image.
- `FLASH_LOCK_UNKNOWN`: Devices MUST NOT return this value.

[GMS-9.5-006] If their system images are preloaded from the factory, Android DEVICEs MUST be in `FLASH_LOCKED` state when they're turned on for the first time out of the box.

9.6 Safe mode

[GMS-9.6-001] Android DEVICEs MUST support booting into safe mode. For example, Google Nexus device users can reboot into safe mode by tapping and holding the power button for 5 seconds.



9.7 DNS privacy

[GMS-9.7-001] DEVICEs running Android 9 or higher MUST support DNS over TLS (https://android-developers.googleblog.com/2018/04/dns-over-tls-support-in-android-p.html) and MUST provide the UI affordance to select the DNS Privacy mode setting from one of **Off**, **Automatic**, or **Private DNS provider hostname**, as defined in AOSP.

Start new requirements for 12

~~**[GMS-9.7-002]** When a DEVICE is set up out of the box, its default DNS Privacy mode setting SHOULD be **Automatic**.~~

[GMS-9.7-002] DEVICEs launching with Android 11 on or after October 1, 2021, and NEW DEVICEs launching with Android 12 or higher, if they are set up out of the box or factory reset by the user, MUST set the DNS Privacy mode to `Automatic` by default.

[GMS-9.7-003] DEVICEs upgrading from Android 11 or lower to Android 12 or higher, if they are factory reset by the user, MUST re-initialize the DNS privacy mode to `Automatic` by default.

Note: The requirement is updated from SHOULD to MUST.

End new requirements

9.8 Data storage encryption

[GMS-9.8-001] From May 1, 2019 onward, DEVICEs launching with Android 9 or higher, if they have more than 512 MB RAM, MUST enable the data storage encryption by default when the user completes the out-of-box setup experience. As per the updated Android 9 CDD (https://source.android.com/compatibility/9/android-9-cdd#9_9_data_storage_encryption), DEVICE implementations with AES performance at or below 50 MiB/sec MAY use Adiantum (https://source.android.com/security/encryption/adiantum) to satisfy this requirement.

9.9 End-to-end encrypted backup

The GMS backup feature running Android 9 or higher supports end-to-end encrypted backup. This feature encrypts user data before sending it to Google's servers. Encryption is done with a key that can only be accessed by knowing a user's secure lock screen credential. This feature supports the reference lock screen implementations in AOSP: pin, pattern, and password.

[GMS-9.9-001] If DEVICEs running Android 9 or higher support a custom primary lock screen mechanism that is different from any of the reference lock screen implementation in AOSP, they MUST inform the users upon setting this custom primary lock screen that their backup data won't be protected with their lock screen credential.

9.10 Passpoint

~~**[GMS-9.10-001]** DEVICEs launching with Android 10 or higher, if they support `FEATURE_WIFI_PASSPOINT`, MUST support EAP-TTLS authentication and the SOAP-XML DEVICE provisioning protocol as defined in the Wi-Fi Alliance Passpoint R2. The DEVICE MUST support user control of provisioning through the Wi-Fi picker.~~

~~**[GMS-9.10-002]** DEVICEs launching or upgrading to Android 11 MUST support Wi-Fi Passpoint (https://www.wi-fi.org/discover-wi-fi/passpoint) and enable the `FEATURE_WIFI_PASSPOINT`. The device MUST:~~

~~* Support EAP-TTLS authentication and the SOAP-XML device provisioning protocol as defined in the Wi-Fi Alliance Passpoint R2. * Process the AAA server certificate as described in Hotspot 2.0 R3 specification. * Support user control of provisioning through the Wi-Fi picker.~~

9.11 Wi-Fi MAC address randomization

[GMS-9.11-001] DEVICEs launching with Android 11 or higher, and with `FEATURE_WIFI` support, MUST randomize the source MAC address used for all STA communication to an Access Point (AP) while associating and associated. The device MUST use a different randomized MAC address for each SSID (FQDN for Passpoint) it communicates with. The device MUST provide the user with an option to control the randomization per SSID (FQDN for Passpoint) with nonrandomized and randomized options, and MUST set the default mode for new Wi-Fi configurations to be randomized. However, OEMs MAY set the default mode to be nonrandomized for the SSIDs of well-known carrier Wi-Fi networks, if the networks rely on factory MAC address for Wi-Fi provisioning.

[GMS-9.11-002] DEVICEs upgrading to Android 10 or higher, and with `FEATURE_WIFI` support, SHOULD randomize the source MAC address used for all STA communication to an AP per the above paragraph.

Start new requirements for 12

~~**[GMS-9.11-003]** DEVICEs upgrading to Android 10 or higher, and with `FEATURE_WIFI_DIRECT` support, SHOULD randomize the source MAC address for all newly formed Wi-Fi Direct connections.~~

Note: This requirement is moved to CDD for Android 12.

End new requirements

[GMS-9.11-004] DEVICEs launching with Android 11 or higher, and with `FEATURE_WIFI` support, MUST use a random BSSID for any AP that they create. The MAC address must be randomized and persisted per SSID used by the AP. The DEVICE MAY provide the user with an option to disable this feature. If such an option is provided, randomization MUST be enabled by default.

Start new requirements for 12

~~**[GMS-9.11-005]** DEVICEs launching or upgrading to Android 11 or higher, and with `FEATURE_WIFI` support, MUST randomize the source MAC address and sequence number of probe request frames, once at the beginning of each scan, while STA is disconnected. Each group of probe request frames comprising one scan MUST use one consistent MAC address (should not randomize MAC address halfway through a scan). Probe request sequence numbers MUST iterate as normal (sequentially) between the probe requests in a scan. Probe request sequence numbers MUST randomize between the last probe request of a scan and the first probe request of the next scan.~~

Note: This requirement is moved to CDD for Android 12.

End new requirements

[GMS-9.11-006] DEVICEs launching or upgrading to Android 11 or higher, and with `FEATURE_WIFI` support, MUST randomize the source MAC address of each IEEE 802.11u access network query protocol (ANQP) message while disconnected.

[GMS-9.11-007] DEVICEs launching or upgrading to Android 11 or higher, and with `FEATURE_WIFI` support, MUST randomize the IEEE 802.11u Generic Advertisement Service (GAS) dialog token in each GAS message.

Start new requirements for 12

[GMS-9.11-008] DEVICEs launching with Android 12 or higher, and with FEATURE_WIFI support MUST allow network suggester apps to enable enhanced MAC randomization on networks they create using the `WifiNetworkSuggestion.Builder#setMacRandomizationSetting(RANDOMIZATION_NON_PERSISTENT)` API. When enhanced MAC randomization is enabled, the MAC address MUST be re-randomized at least every 24 hours but not while continuously connected. The randomized MAC MUST NOT be persisted and MUST be different on every re-randomization. If the device is connected at the 24 hour mark then it SHOULD re-randomize it's MAC address before it's next connection to an AP.

Additionally, if a network suggestion gets removed and then is later added again with enhanced MAC randomization enabled, the MAC address for that network MUST be immediately re-randomized.

[GMS-9.11-009] DEVICEs upgrading to Android 12 or higher, and with FEATURE_WIFI support SHOULD allow network suggester apps to enable enhanced MAC randomization as per [GMS-9.11-008].

End new requirements

9.12 App Integrity Manager

Android 11 introduces the App Integrity Manager, which helps app developers protect the integrity of their apps. It relies on the offline app integrity rule configuration file.

[GMS-9.12-001] DEVICEs running Android 11 or higher MUST:

- Keep the AOSP implementation of the App Integrity Manager unchanged.
- Allow the trusted rule provider to update the offline app integrity rule configuration file.
- Keep GMS Core listed as the trusted rule provider and the Google Play Store as the backup trusted rule provider.

9.13 Wi-Fi IMSI privacy protection

[GMS-9.13-001] DEVICEs launching or upgrading to Android 11 or higher MUST obtain user consent before connecting to a SIM-based carrier Wi-Fi network that doesn't support IMSI privacy protection. AOSP provides a reference implementation that's compliant to this requirement.

Start new requirements for 12

9.14 Domain verification asset link enforcement

[GMS-9.14-001] DEVICEs launching or upgrading to Android 12 or higher MUST use GMS Core as the only Google Digital Asset Links (https://developers.google.com/digital-asset-links/v1/getting-started) handler for verifying app link domains (https://developer.android.com/training/app-links/verify-site-associations).

9.15 Safety and Emergency settings

[GMS-9.15-001] On DEVICEs running Android 12 or higher, the **Safety & Emergency** section MUST be a top-level item in the **Settings** app, as implemented in the AOSP.

[GMS-9.15-002] The submenu in the **Safety & Emergency** section MUST feature the following items.

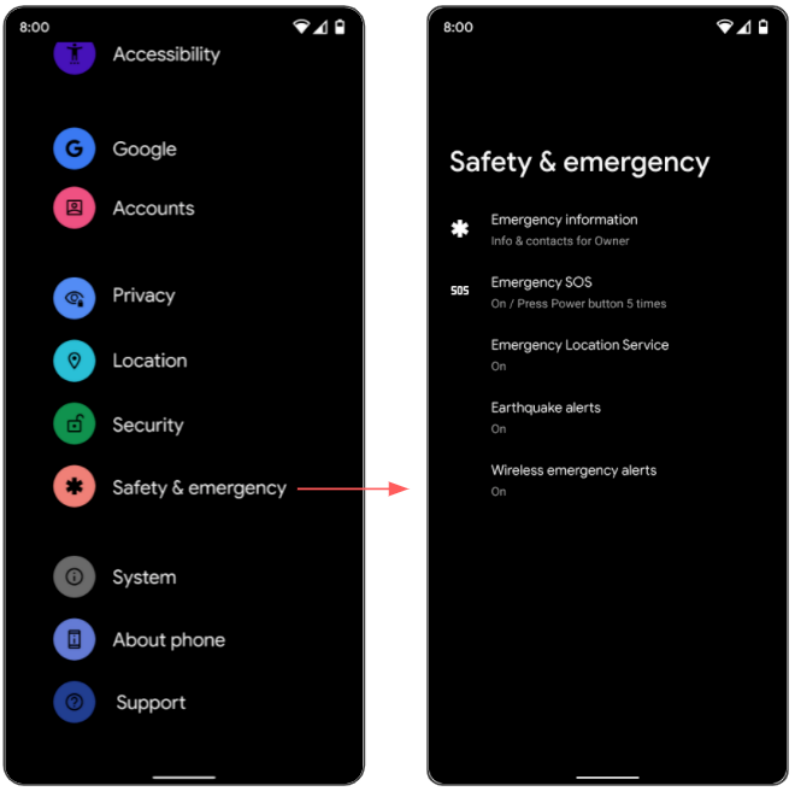
- Emergency Information: This item MUST allow users to enter their medical information and emergency contacts information, which MAY be optionally displayed on the lockscreen. It MAY be separated into two separated items as **Medical information** and **Emergency contacts**.
- Emergency SOS: This item MUST be linked to the Emergency SOS feature per Emergency SOS experience (/gms/policies/domains/protect-users#sos-exp).

[GMS-9.15-003] Partners MUST allow the Google Play services app to inject the following three items into the **Safety & Emergency** section, as implemented in the AOSP.

- Emergency Location Service
- Earthquake Alerts
- Wireless Emergency Alerts

Note: The Google Play services app MAY NOT inject all three items depending the availability of services per user's location.

[GMS-9.15-004] Partners MAY include additional Safety & emergency features in the submenu of **Safety & Emergency** section.



Emergency SOS experience

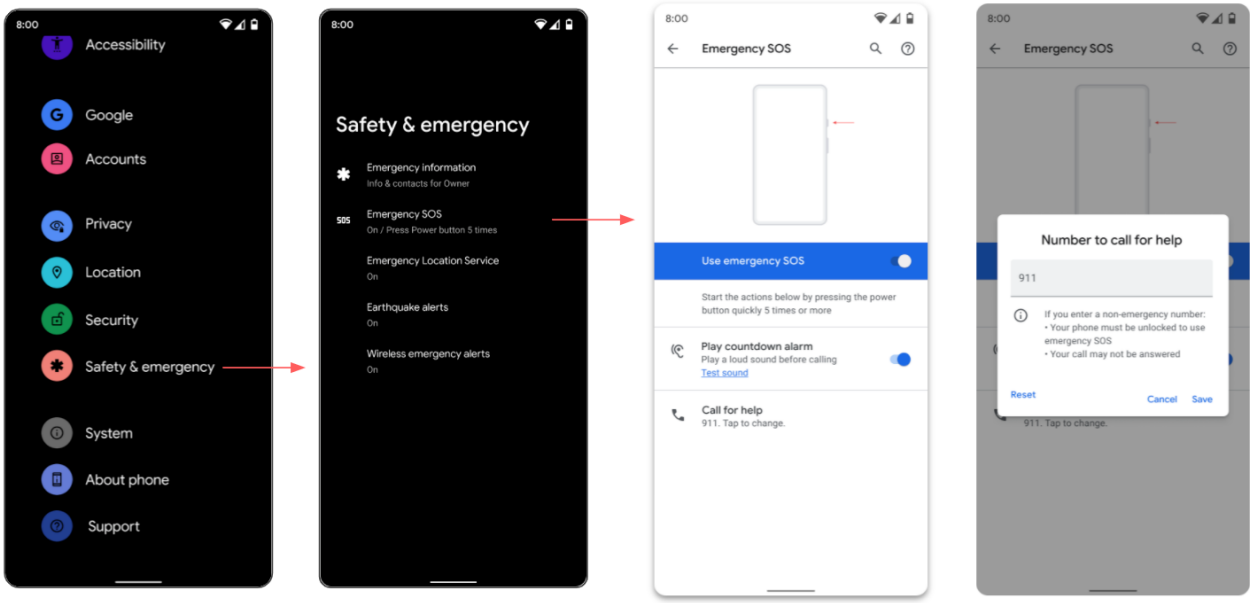
[GMS-9.15-005] DEVICEs running Android 12 or higher MUST include the Emergency SOS experience. The Emergency SOS MUST be placed in **Settings** > **Safety & Emergency**.

[GMS-9.15-006] This feature is STRONGLY RECOMMENDED to meet the below requirements:

- The setting should be **ON** by default for emergency number dialing (optional services excluded).
- Press pKey 5x times (unless local regulations require otherwise) to trigger an emergency experience.
- Trigger a 5-second countdown with alarm sound before triggering emergency call.
- Allow users to trigger an emergency call (112, 911, 999, and so on.)
- Provide an option for users to change the designated emergency number by tapping and manually entering a new number.

In addition,

- Partners MAY provide additional services such as emergency sharing to emergency contacts, photos, videos or sound recordings.
- Users MAY opt-out of the experience by going to **Settings** > **Safety & emergency** > **Emergency SOS**.

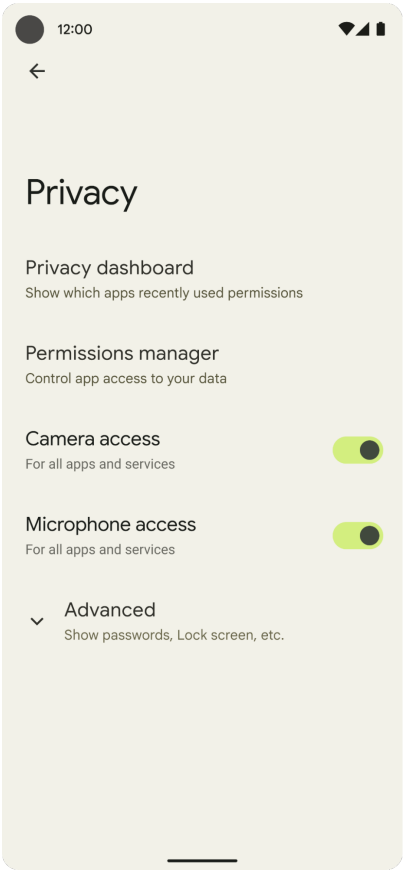


Emergency SOS for AOSP | Change default number

9.16 Privacy dashboard

[GMS-9.16-001] On DEVICEs running on 11 or higher, if partners implement a privacy dashboard displaying the use of access to APIs protected by the dangerous (runtime) permissions (https://developer.android.com/reference/android/Manifest.permission), then:

- The Privacy dashboard MUST be placed within the Privacy Settings page (`ACTION_PRIVACY_SETTINGS` intent).
- The `ACTION_REVIEW_PERMISSION_USAGE` intent MUST invoke the privacy dashboard settings activity.



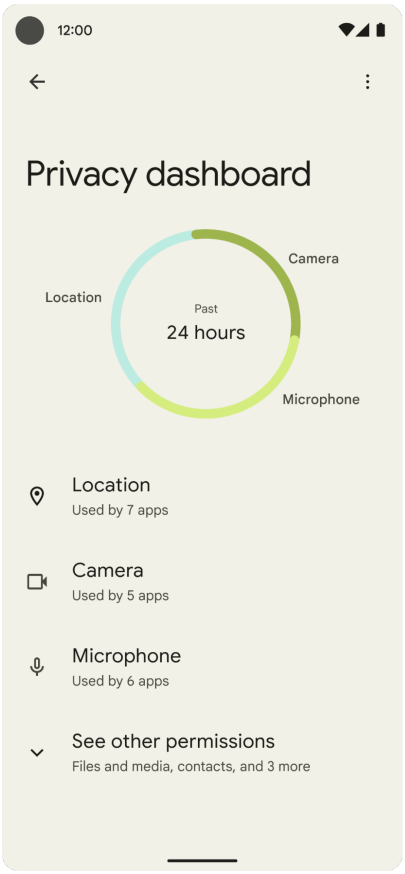
- The privacy dashboard MUST only use the `AppOpsManager` APIs to read access history of Android APIs protected by the dangerous (runtime) permissions (https://developer.android.com/reference/android/Manifest.permission). The privacy dashboard MUST not create an alternate repository than the one managed by the `AppOpsManager` or manipulate the access history of Android APIs protected by dangerous (runtime) permissions (https://developer.android.com/reference/android/Manifest.permission).
- The privacy dashboard MUST display only the access history of APIs protected by all dangerous (runtime) permissions (https://developer.android.com/reference/android/Manifest.permission) for the devices' API level.
- The privacy dashboard MUST ONLY display the following access history:
  - Timeline of every distinct access with timestamp for the last 24 hours and/or 7 days for the access of APIs protected by the following dangerous permissions:
    - `RECORD_AUDIO`
    - `CAMERA`
  - Timeline of every distinct access, except for access events logged with the attributionTag (https://developer.android.com/guide/topics/data/audit-access) declared by the location provider(s) (with the

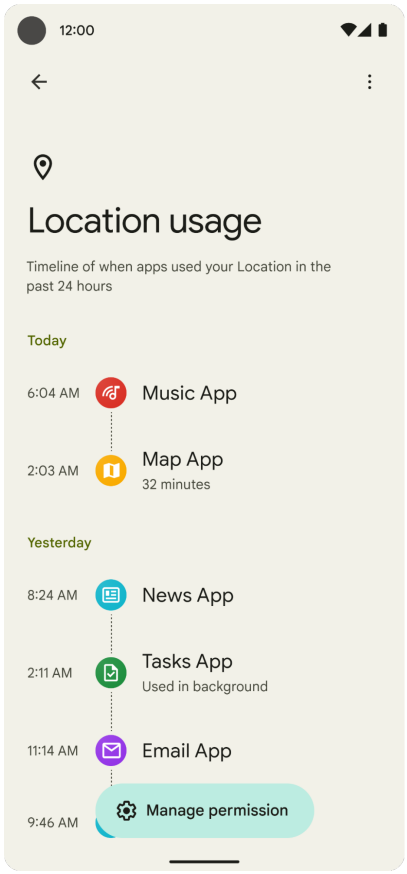
`android:location_allow_listed_tags` in the provider service declaration) in accordance with Location access (/gms/policies/domains/protect-users#loc-access), for the last 24 hours and/or 7 days for the access of APIs protected by the following dangerous permissions:

- `ACCESS_FINE_LOCATION`
- `ACCESS_COARSE_LOCATION`

- Timeline of every distinct access, except for access events logged with the `attributionTag` (https://developer.android.com/guide/topics/data/audit-access) declared by an activity recognizer(s) with the metadata `android:activity_recognition_allow_listed_tags` list in accordance with Location access (/gms/policies/domains/platform-policies#loc-access), for the last 24 hours and/or 7 days for the access of APIs protected by the following dangerous (runtime) permissions (https://developer.android.com/reference/android/Manifest.permission):
  - `ACTIVITY_RECOGNITION`
- Last distinct access within the last 24 hours and/or 7 days for all dangerous (runtime) permissions (https://developer.android.com/reference/android/Manifest.permission) for the devices' API level, except for the ones listed above.
- Data that can be verified by the system such as package name and the permission accessed.

- The privacy dashboard MUST show an extra user affordance for any app, and tapping it MUST invoke an activity that handles the `VIEW_PERMISSION_USAGE_FOR_PERIOD` intents.
- Accesses that are in subsequent minutes by the same APK MUST be grouped into a continuous duration.
- No apps MUST be hidden in the Permissions history page or all system apps overflow menu. In the Permissions history page, there should be an overflow menu that allows users to see system apps. ALL system apps MUST be placed in the all system apps overflow menu. The system apps includes:
  - Pre-installed app AND
  - Headless (no visible launcher) app AND
  - Pre-granted permission





9.17 Package Installer app

[GMS-9.17-001] DEVICEs MUST preload only one Package Installer app, which handles `ACTION_INSTALL_PACKAGE` intent, and it MUST be the Google-signed version of AOSP Package Installer APK (`GooglePackageInstaller.apk`) that is included in the GMS package.

9.18 Activity Recognizer

Activity Recognizer is an API provider which allows Android apps to detect the user's current activity (for example, walking, driving, running), by combining the information from various on-device sensors.

[GMS-9.18-001] Every preloaded Activity Recognizer on DEVICEs running Android 12 or higher:

- MUST be a system AR role holder by adding its package name to the `config_systemActivityRecognizer` framework resource.

- MUST have a service component which resolves the `ACTIVITY_RECOGNIZER` intent.
- MUST protect the access to its API with the `ACTIVITY_RECOGNITION` permission.

9.19 Deprecate insecure VPN types

[GMS-9.19-001] DEVICEs launching or upgrading to Android 12 or higher on or after Dec 9, 2021 MUST NOT allow creation of new VPN configurations of any of the following types:

- PPTP
- L2TP/IPSec PSK
- L2TP/IPSec RSA
- IPSec Xauth PSK
- IPSec Xauth RSA
- IPSec Hybrid RSA

End new requirements

10 Android Go

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

Requirements	Links to Domains
10	Introduction - 2GB of RAM Strongly recommended (/gms/policies/domains/go#go-intro)
10.2	Core Services (/gms/policies/domains/go#core-services)
10.2	Optional GMS Apps (Default Assistant role holder) (/gms/policies/domains/go#optional-gms-apps)

Introduction

Android CDD defines *a class of devices with limited size of RAM* (low-memory device) and requires them to return `true` for the public API `ActivityManager.isLowRamDevice()` (https://developer.android.com/reference/android/app/ActivityManager#isLowRamDevice()).

After October 20, 2020 DEVICEs MUST be defined as a Go device, with the `isLowRamDevice` flag set to `true`, if the DEVICE meets **both** of these criteria:

- The DEVICE launches with Android 10 or higher.
- The DEVICE has 2 GB of RAM or less.

Start new requirements for 12

Beginning with Android 12, 2 GB of RAM is **STRONGLY RECOMMENDED** for new Go devices.

End new requirements

A 2 GB RAM device that launched as a standard DEVICE (non-Go) SHOULD NOT convert to an Android Go device using MRs or letter upgrades.

A 3 GB or greater RAM device MAY be approved as a Go PRODUCT variant, but only in conjunction with a Go device with 2 GB RAM or less. A device with more than 2 GB RAM CANNOT be a Go PRODUCT alone in a DEVICE group.

Beginning with Android 11, devices with 512 MB RAM (including upgrades) aren't qualified for preloading GMS. Previously launched 512 MB RAM devices shouldn't upgrade to

Android 11.

Partners are required to sign the *MADA addendum for Android Go* to ship the GMS on low-memory devices and launch with Android 8.0 or higher. Such devices (GMS Go devices) MUST meet the requirements in this section, which override some conflicting requirements in other sections in this document.

In this section, all references to memory sizes, for example 512 MB, refer to the binary notation of megabyte (1024x1024).

Note: This section doesn't apply to DEVICEs that launched with Android 7.x or lower that upgraded to Android 8.0 or higher.

10.1 Geo-availability for Android Go

[GMS-10.1-001] GMS Go devices MUST be compliant with the *Geo-availability for Android Go*, which is available in Geo-availability (/gms/policies/overview/geo-availability#android-go). If a GMS app isn't listed in the Geo-availability for Android Go, its standard geo-availability (/gms/policies/overview/geo-availability) MUST be applied.

10.2 GMS Go Core services and apps

GMS Go Core services and apps include required apps and commonly used services. The required apps are listed below and must be presented with launcher icons as shown in Home screen appearance (#home-screen-appearance).

Core services

[GMS-10.2-001] GMS Go devices MUST preload the complementary Core services listed below along with the Core services listed in Core services and apps (/gms/policies/domains/best-google-experience#core-services-and-apps).

APK filename/package name	Description	Note
LatinImeGoogleGo	Lightweight	This version shares the same package

`com.google.android.inputmethod.latin`	version of Gboard app for GMS Go devices.	name with the full version but declares `android.hardware.ram.low` feature flag in its manifest.
NavGo `com.google.android.apps.navlite`	A headless app that implements turn-by-turn navigation feature for the Maps Go app.	From November 15, 2018 onward, Go devices that preload Maps Go app MUST also preload NavGo. Go devices that use the full version of Maps aren't required to preload NavGo, but there's no functional impact if preloaded anyway.

Start new requirements for 12

APK filename/package name	Description	Note
SearchSpeechServices `com.google.android.apps.speechservices`	Supports Google speech recognition service on GMS Go devices.	For DEVICEs launching on Android Go 12 or higher, the `SearchSpeechServices` core service **MUST NOT** be preloaded. DEVICES upgrading to Android 12 or higher **MUST** remove the core service during upgrade. For DEVICEs launching or upgrading to Android Go 11 or lower, the `SearchSpeechServices` core service **MUST** be preloaded.

End new requirements

Core apps (subject to geo-availability)

[GMS-10.2-002] The following apps replace the Core apps, defined in Core services and apps (/gms/policies/domains/best-google-experience#core-services-and-apps), for GMS Go devices.

512 MB RAM	1 GB RAM	>=2 GB RAM
Google Assistant Go	Google Assistant Go	Google Assistant Go
Google Chrome	Google Chrome	Google Chrome
Gallery Go	Gallery Go	Gallery Go
Gmail Go	Gmail Go	Gmail (full version)
Google Go	Google Go	Google Go
Maps Go	Maps Go	Maps (full version)
Play Store	Play Store	Play Store
YouTube Go [1]	YouTube (full version)	YouTube (full version)
Google Duo	Google Duo	Google Duo
		Google Drive
		Play Movies
		YouTube Music

1 Include YouTube Go or YouTube (full version) depending on the geo-availability rules.

Note the following core app requirement changes.

Effective date	Requirement change
September 3, 2019	On Go devices launched on or after September 3, 2019, Maps Go is a required core app.
November 4, 2019	On Go devices launched on or after November 4, 2019, Gallery Go is a required

	core app.
January 2, 2020	Go devices launching on or after January 2, 2020 MUST provide a digital wellbeing solution with parental controls at the top level of the settings app per **Digital wellbeing and parental controls** (#ditigal-wellbeing-and-parental-controls). Go devices that launched before January 2, 2020 are STRONGLY RECOMMENDED to provide a digital wellbeing solution with parental controls at the top level of the settings app.
February 3, 2020	On Go devices launched on or after February 3, 2020, Google Duo is a required core app. For Go devices with 512 MB RAM, Duo is optional.
June 24, 2020	On Go devices launched on or after August 24, 2020, 1 GB devices MUST use the full version of YouTube.

Optional GMS apps

[GMS-10.2-003] In addition to the required Go core apps listed above, partners MAY distribute any other GMS apps. GMS apps not listed in the Core apps requirements are considered optional, and count toward the app preload quota. Optional GMS apps are subject to Geo-availability rules.

Start new requirements for 12

Default Assistant role holder on Android Go devices

[GMS-10.2-004] Go DEVICEs launching or upgrading to Android 12 or higher MUST:

- Preload the latest release of **Google Search Go app** (i.e. version 3.32.381795618.31release.go or higher) and **Assistant Go app** (i.e. 2.10.0.380528351.armeabi-v7a.31release or higher) in the system image.
- Set the Google Search Go app as the default assistant role holder, by setting the config_defaultAssistant framework resource overlay with the package name of the Google Search Go app.

```
<string name="config_defaultAssistant" translatable="false">
com.google.android.apps.searchlite </string>
```

End new requirements

10.3 Home screen appearance

The default home screen for GMS Go devices MUST feature the following items.

Placement of the Google search widget

[GMS-10.3-001] To provide a consistent user experience, the approved implementation of the Google Search Widget is provided by the Google Go app.

Placement of the Google collections folder

[GMS-10.3-002] The Google collections folder on the default home screen:

- MUST be labeled "Google."
- MUST contain app icons for all preloaded Core GMS apps on the device.
- MUST NOT include any non-Google apps.

Note: On Go devices launching on or after December 2, 2019, the default GMS bundle app icon order in the Google folder is preferred, but partners can rearrange the icons to any order if needed.

Placement of the Play Store app icon

[GMS-10.3-003] The default home screen MUST have a Play Store icon.

Hotseat layout

[GMS-10.3-004] The hotseat layout:

- MUST contain at least four app icons.
- MAY contain five app icons on a device with a display HD 720 p or above.

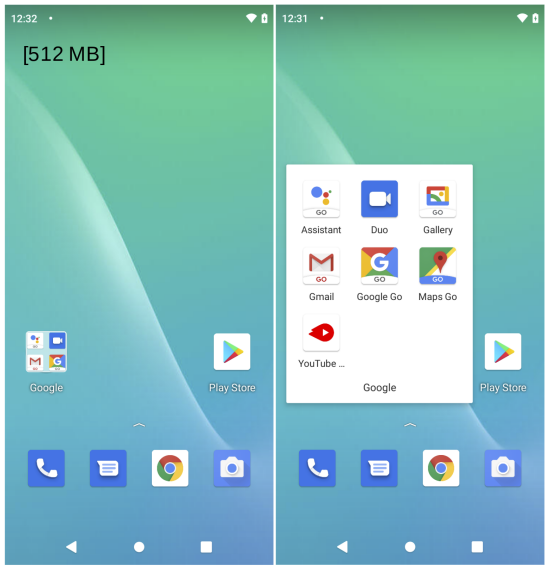
Placement on Apps menu

[GMS-10.3-005] For Go devices with an Apps menu:

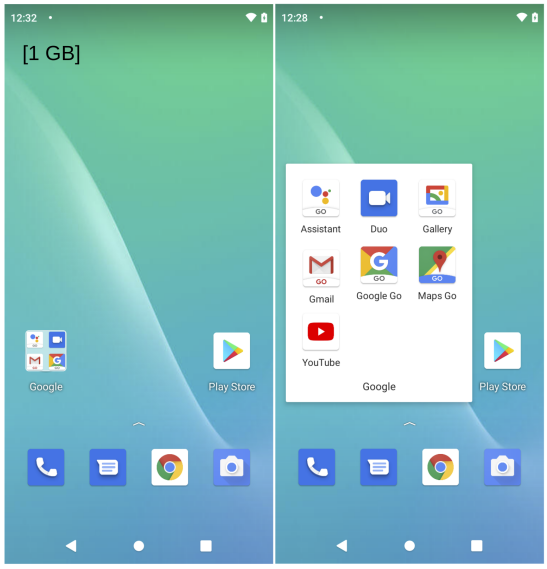
- All GMS apps MUST be placed in the Apps menu.
- The Play Store MUST NOT reside within any folder on the Apps menu and SHOULD be in the top level of the Apps menu.

Example home screen layout

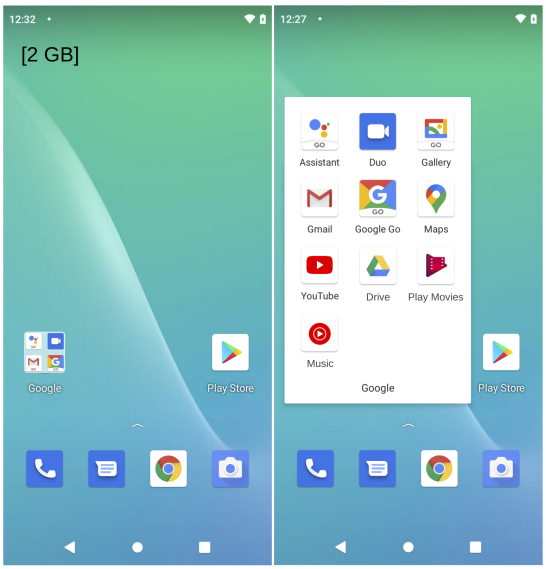
512 MB RAM



1 GB RAM



2 GB RAM



The default home screen for GMS Go devices MAY allow the following options.

Placement of additional collections folders

[GMS-10.3-006]

- Devices with less than 2 GB of RAM, Partners MAY choose up to two additional apps or folders to be placed directly above the hotseat.
- Devices with 2 GB of RAM don't have any app or folder limitations.

Placement of GMS apps in hotseat or default home screen

[GMS-10.3-007] Any GMS app MAY be placed in the hotseat or on the default home screen. If a GMS app is placed in the hotseat or the default home screen, then the app icon SHOULD be removed from the Google collections folder.

Additional home screens

[GMS-10.3-008] The default home screen for Go devices MAY feature additional home screens based on the rules below:

- Devices with less than 2 GB of RAM:
  - If the launcher app contains an Apps menu, then additional home screens placed by a partner are NOT allowed.

    - Otherwise, additional home screens MAY be added to the right of the default home screen by the partner.
- Devices with 2 GB of RAM, additional home screens MAY be added to the right of the default screen by the partner.

Note: Partners SHOULD limit the total number of home screens to no more than two for the best out of box experience.

Note: Phone users are allowed to add their own additional home screens if they choose.

Folders in the Apps menu

[GMS-10.3-009] If a partner chooses to implement folders within the Apps menu (if applicable):

- The GMS app icons and GMS Go Core app icons SHOULD be combined within a Google folder.
- Alphabetical order of the GMS app icons in the Google folder is preferred. Optionally partners can decide to use a different app icon order.

10.4 User data partition

[GMS-10.4-001] For optimal performance, the F2FS filesystem MUST be enabled on the user data partition for flash-based storage devices.

[GMS-10.4-002] The user data partition allows users to install apps and store content on their device. The user data partition size MUST meet or exceed the minimums listed below.

Flash storage size	User data partition minimum
4 GB	1.2 GB
8 GB	5 GB

16 GB	12 GB
>= 32 GB	24 GB

Note: If a GMS Go Device uses a file based swap outside of the user data partition, the size of the user data partition can be reduced by 0.2 GB for 512 MB RAM devices and by 0.4 GB for 1 GB RAM devices.

10.5 RAM usage

[GMS-10.5-001] To provide a good user experience on low RAM devices, limits are set for free memory on boot, and for persistent processes (total PSS).

Preparing a device memory measurement

To calculate a device's free memory or to calculate a device's persistent memory, setup the device by ensuring that:

- Apps are only preinstalled by the device manufacturer. No additional side-loaded or Play Store downloaded apps are permitted on the device.
- Device settings are in the default out-of-box state.
- After first boot and onboarding, the device is rebooted and allowed to rest for at least 5 minutes.
- During measurements, the screen remains on and the device remains unlocked.

Device free memory on boot

Depending on a device's configured screen size, the PSS value of `MemAvailable` in `cat/proc/meminfo` and the dirty PSS of cached processes at warm boot should meet or exceed the following.

Device RAM size	PSS memory available
512 MB	125 MB

768 MB	285 MB
1 GB	430 MB
1.5 GB	650 MB
2 GB	850 MB

Calculate the device's free memory:

1. To get `MemAvailable` values, run:

```
$ adb shell cat /proc/meminfo
```

2. Identify all the cached processes:

```
$ dumpsys meminfo
```

3. Sum **private dirty** and **private clean** memory of each process identified in step 2.
4. Add the results in step 1 and step 3.

Persistent processes total memory usage

The persistent processes total PSS must not exceed the values in the table. This is calculated using `dumpsys meminfo`, and looking at the Total PSS by OOM adjustment under the persistent header.

Screen size	512 MB RAM	1 GB RAM	2 GB RAM or higher
HVGA	80 MB	85 MB	100 MB
(F)WVGA	85 MB	90 MB	110 MB

(F)WVGA+			
qHD	N/A	95 MB	120 MB
HD or higher	N/A	110 MB	130 MB

To calculate a device's persistent memory:

1. Enter the command:

```
$ adb shell dumpsys meminfo
```

2. Persistent memory is the **Total PSS by OOM adjustment** under the **persistent** header

10.6 Display and camera resolution limits

[GMS-10.6-001] To provide a good user experience and good performance on low RAM devices, the display resolution and camera resolution MUST NOT exceed the limits below.

RAM size	Maximum display resolution	Maximum camera resolution
512 MB	(F)WVGA+	5 MP
> 512 MB	No limit	No limit

10.7 App preload

Definition of headed app

An app that registers for the following intent filter is considered a headed app:

```
<intent-filter>
   <action android:name="android.intent.action.MAIN" />
   <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Number of preloaded headed apps

- **[GMS-10.7-001]** The maximum number of headed apps on the device preloaded, or otherwise installed by device manufacturer without explicit user opt-in, MUST NOT exceed the limits below.
- Core apps defined in [GMS-10.2-002], and one app from each of the exempt headed app categories listed below don't count toward the preload limit.

RAM size	Flash size (GB)	Max # of headed apps	Change history
< 2 GB	< 16	9	7 if launched before November 4, 2019 8 if launched before January 2, 2020
< 2 GB	>= 16	System: 9 User data: 9	7+7 if launched before November 4, 2019 8+8 if launched before January 2, 2020
2 GB	< 16	15	
2 GB	>= 16	System: 15 User data: 15	

Exempt headed app categories

If a category below doesn't have a default app, then one app per category is exempt, and the rest count toward the limit of preloaded headed apps.

For DEVICEs launching on or after November 4, 2019, if you preload your own gallery app, then it's exempt and doesn't count toward the limit of preloaded headed apps.

App category	Identifying the exempt app

Calculator	Action: `ACTION_MAIN` Category: `CATEGORY_APP_CALCULATOR`
Calendar	Action: `ACTION_MAIN` Category: `CATEGORY_APP_CALENDAR`
Camera	Provider for intent with Action: `ACTION_IMAGE_CAPTURE`
Clock	Action: `ACTION_SHOW_ALARMS` Category: `CATEGORY_APP_DEFAULT`
Contacts	Action: `ACTION_MAIN` Category: `CATEGORY_APP_CONTACTS`
Dialer	Provider for intent with action: `ACTION_DIAL`
Files	Action: `android.os.storage.action.MANAGE_STORAGE` Category: `android.intent.category.DEFAULT`
FM radio	Verification through allow list. Work with your Google Account Manager to add your FM radio app to the allow list.
Gallery	Action: `ACTION_MAIN` Category: `CATEGORY_APP_GALLERY`
SMS	Action: `ACTION_MAIN` Category: `CATEGORY_APP_MESSAGING`
Music player	Action: `ACTION_MAIN` Category: `CATEGORY_APP_MUSIC`
Launcher	Action: `ACTION_MAIN` Category: `android.intent.category.HOME`
Settings	Action: `android.settings.SETTINGS` Category: `android.intent.category.DEFAULT`
SIM toolkit	Verification through allow list. Work with your Google Account Manager to add your SIM toolkit app to the allow list.

Sound recorder	Verification through allow list. Work with your Google Account Manager to add your Sound recorder app to the allow list.
Weather	Verification through allow list. Work with your Google Account Manager to add your Weather app to the allow list.

10.8 App memory

[GMS-10.8-001] All preloaded apps and all apps installed by OEMs without user action on the devices MUST meet the performance requirements outlined below. This includes Google apps, all apps installed by OEMs without user action, apps requested or required by a telecom operator and apps installed using OTA/PAI mechanisms.

Peak PSS memory usage requirements

On a clean device, depending on the configured screen size, preloaded apps and apps installed by OEMs in each app category below MUST not exceed the peak PSS memory usage requirements outlined below.

To test each app's peak PSS memory usage requirement, first setup a device by ensuring that:

- Apps on the device represent only those preinstalled by the device manufacturer.
- The settings should be in the default out-of-box state in which the device will be shipped.
- The device is connected to a stable Wi-Fi and includes a working SIM card.
- The screen is on and the device is unlocked when measuring values.
- All other apps are closed before starting a test. Other background apps can negatively affect the test cases.

The following table describes how to test each app category, and it lists the peak PSS usage in MB.

App category	Test scenario steps	< 2 GB RAM	2 GB RAM

App category	Test scenario steps	< 2 GB RAM	2 GB RAM
Dialer Default	1. Open the phone app. 2. Make a phone call by dialing a number. 3. Hang up. 4. Close the dialer app, and exit to the home screen.	90	150
Browsers All preloads	1. Open the browser app. 2. Navigate to `about:blank`. Navigating to a website other than `about:blank` can affect test results. 3. Keep the browser open for 5 seconds after the page fully loads. 4. Close the browser app, and exit to the home screen. (Note: PSS targets include the rendered process.)	130	150
SMS app Default	1. Open the messaging app. 2. Send a text message. 3. Close the messaging app, and exit to the home screen.	90	150
Gallery Default	1. Open the gallery app. 2. Open a sample image. 3. View the image for 5 seconds. 4. Close the gallery app, and exit to the home screen. (Note: Allow-listed cloud galleries are exempt from this test.)	90	150
Camera Default	1. Open the camera app. 2. Use the default camera to take a picture. 3. Close the camera app, and exit to the home screen.	90	150
Music app Default	1. Open the music app. 2. Play a music file for 30 seconds. 3. Close the music app, and exit to the home screen.	90	150

App category	Test scenario steps	< 2 GB RAM	2 GB RAM
Streaming music app Default	1. Open the streaming music app. 2. Wait for 5 seconds. 3. Close the streaming music app, and exit to the home screen.	130	150
Launcher Default	1. Reboot the device. 2. Unlock the screen. 3. Stay on the home screen for 60 seconds. 4. Measure the memory usage. (Note: For the launcher, the measurement is taken after the wait time, rather than during the peak, unlike other apps.)	90	150
Streaming video player Default	1. Open the streaming video player app. 2. Wait for 5 seconds. 3. Close the streaming video app, and exit to the home screen.	130	150
Maps	1. Open the maps app. 2. Wait for 10 seconds. 3. Close the maps app, and exit to the home screen.	135	135
All other apps	1. Open the app. 2. Wait for 60 seconds. 3. Grant all permissions and set up the app. 4. Close the app and clear all caches. 5. Start measurements. 6. Open the app. 7. Wait for 5 seconds. 8. Close the app, and exit to the home screen.	90	150

11 Android Enterprise

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Updates in upcoming GMS requirements release

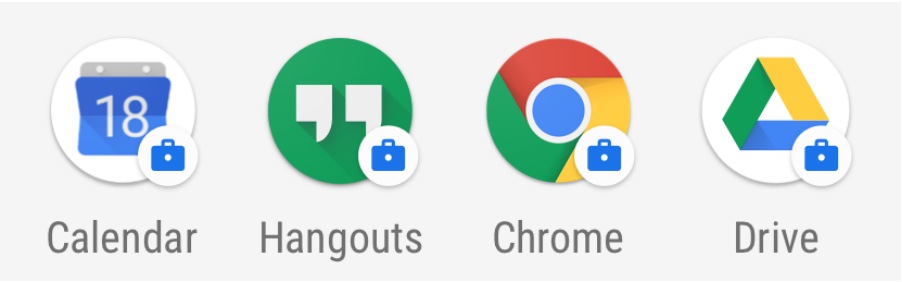
Requirements	Links to domains
11.1.1	Work Profile badging (#GMS-11.1.1-001)
11.1.2	Work Profile user experience (#GMS-11.1.2-001)
11.1.3	Share sheet and app chooser user experience (#GMS-11.1.3-001)
11.1.4	Work profile user education (#GMS-11.1.4-001)
11.3	Setup experience (#GMS-11.3-001)
11.4	Enterprise required apps (#GMS-11.4-001)
11.5	Enterprise default cross-profile apps (#GMS-11.5-001)
11.6	Cross-profile integration requirements (#GMS-11.6-001)

11.1 Work Profile UX consistency requirements

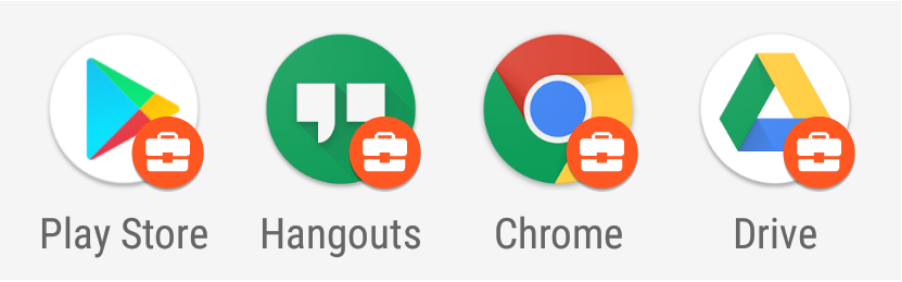
11.1.1 Work Profile badging

[GMS-11.1.1-001] To provide user experience compatibility and minimize support costs for enterprise customers, the design of the badge for managed apps, managed widgets, and other badged UI elements MUST be reviewed and approved by Google, unless the DEVICE implementation uses the exact same asset in AOSP.

Badge for Android 9 or higher



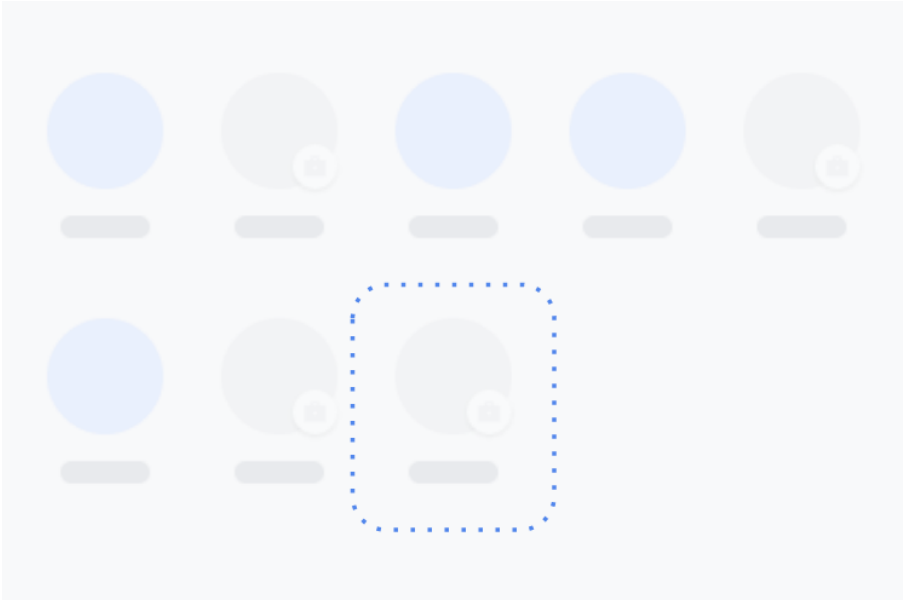
Badge for Android 8.1 or lower



Start new requirements for 12

[GMS-11.1.1-002] In DEVICEs launching or upgrading to Android 11 or higher:

When the Work Profile is off, the work app icons (apps that are part of the Work Profile) MUST have their opacity lowered to between 54% – 70% and displayed in grayscale, as implemented in AOSP.



End new requirements

11.1.2 Work Profile user experience

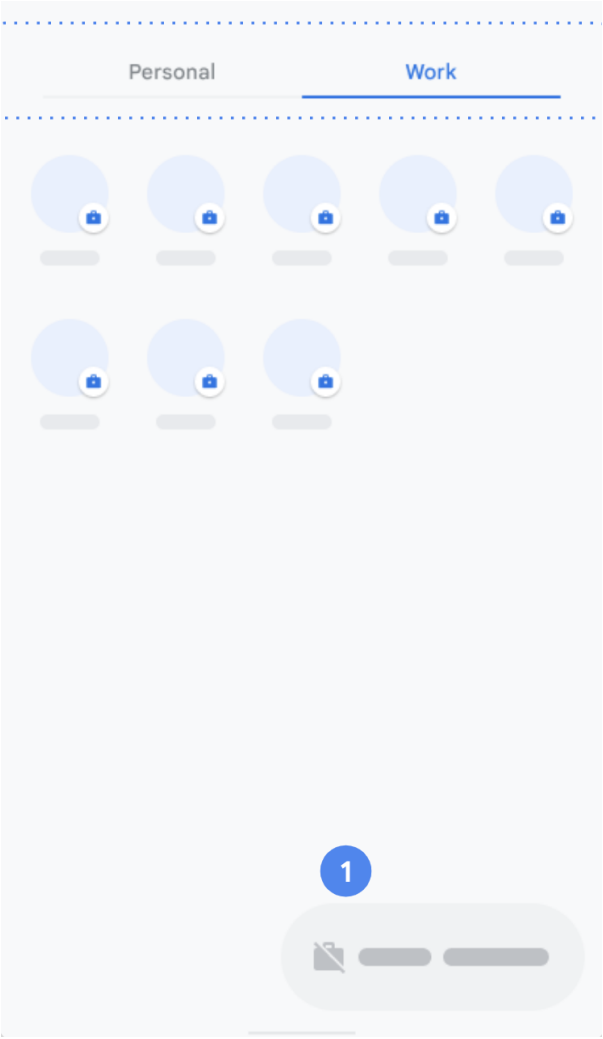
The Work Profile user experience varies based on whether the default launcher has or doesn't have an All Apps tray. These are described below.

Start new requirements for 12

[GMS-11.1.2-001] In DEVICEs launching or upgrading to Android 11 or higher that report `android.software.managed_users` if the **default launcher has an All Apps tray (apps drawer)**:

- **[GMS-11.1.2-001.001]** Work apps MUST be grouped together within the all apps tray using tabs or similar UI components. If tabs are used to separate work apps from personal apps, they MUST be named **Personal** and **Work**.

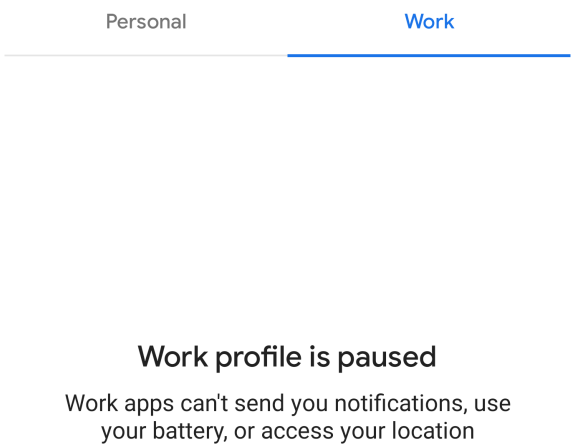
- **[GMS-11.1.2-001.002]** If tabs are used to separate work apps from personal apps, the user SHOULD be able to turn off the Work Profile from the Work tab.



- ~~**[GMS-11.1.2-001.003]** When the Work Profile is off, work app icons (apps that are part of the Work Profile) on the home screen MUST have their opacity lowered to~~

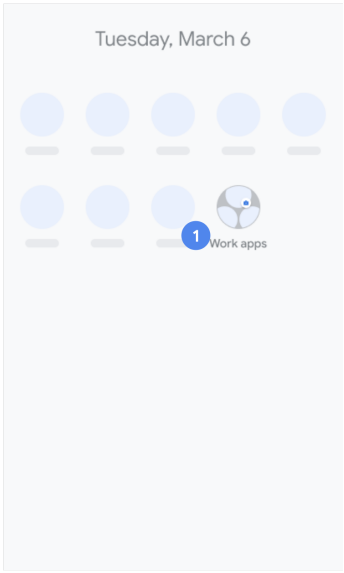
~~between 54%–70% and displayed in grayscale, as implemented in AOSP.~~

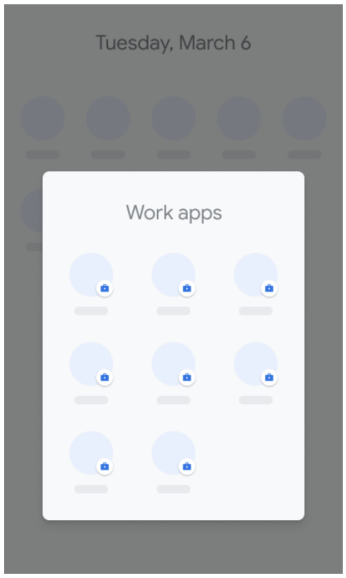
- **[GMS-11.1.2-001.003]** If tabs are used to separate work apps from personal apps, when the Work Profile is off, an overlay on the Work tab SHOULD describe that work apps are paused and all work apps should be hidden.



[GMS-11.1.2-002] In DEVICEs launching or upgrading to Android 12 or higher that report `android.software.managed_users`, if the default launcher does NOT have an All Apps tray (apps drawer):

- ~~**[GMS-11.1.2-002.001]** When the Work Profile is off, work app icons (apps that are part of the Work Profile) MUST have their opacity lowered to between 54%–70% and displayed in grayscale, as implemented in AOSP.~~
- **[GMS-11.1.2-002.001]** Work apps MUST be contained in a folder on the Home screen.





End new requirements

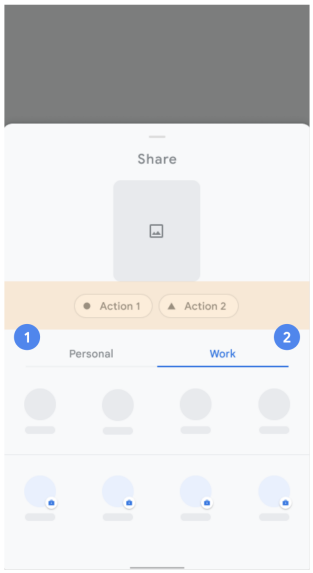
11.1.3 Share sheet and app chooser user experience

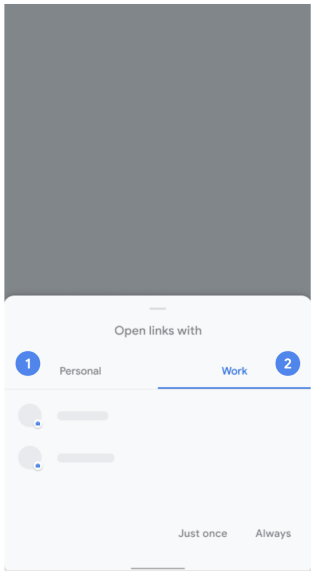
Start new requirements for 12

[GMS-11.1.3-001] In DEVICEs launching or upgrading to Android 11 or higher that report `android.software.managed_users`, share sheet and app chooser implementations MUST separate Work and Personal Profile targets across two tabs and meet the following requirements.

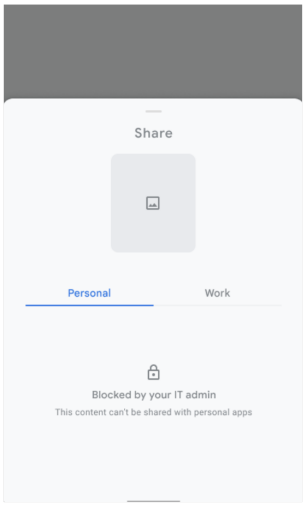
End new requirements

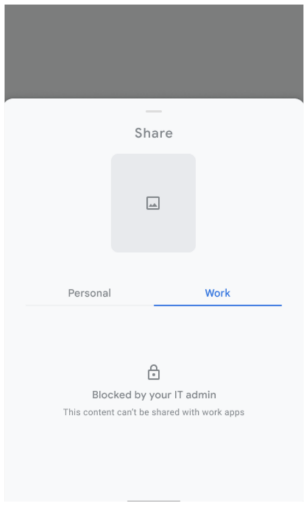
- **[GMS-11.1.3-001.001]** The tabs MUST be named *Personal* and *Work*.
- **[GMS-11.1.3-001.002]** When users switch between Personal and Work tabs, the target objects in each tab MUST be shown immediately, by populating the target objects for both Personal and Work tabs in advance.
- **[GMS-11.1.3-001.003]** If the launcher implements Personal and Work tabs in the all apps tray, the visual design and the user interface of these tabs SHOULD be consistent with those of Personal and Work tabs in the share sheet and the app chooser implementation.
- **[GMS-11.1.3-001.004]** The user MUST land on the appropriate tab when launched (that is, work if from work apps, personal if from personal apps).

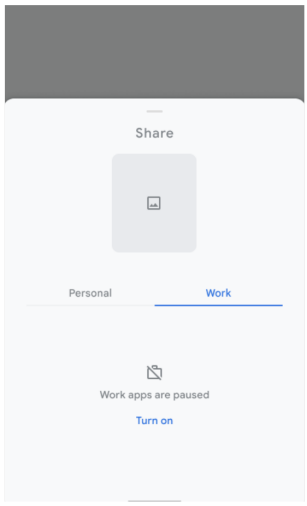




- **[GMS-11.1.3-001.005]** If a profile doesn't contain any valid share targets, the corresponding tab MUST still be present, MUST NOT show any targets, and MUST show a message explaining why there are no targets available to the user (for example, no apps available, action disabled by admin). Refer to AOSP for the reference implementation.







Start new requirements for 12

- **[GMS-11.1.3-001.006]** In DEVICEs launching or upgrading to Android 12 or higher that report `android.software.managed_users`, when a user taps an auto verified link that can't be handled in the current profile, but an app in the other profile can open an app in the other profile that can open that link, then an app chooser MUST be shown.

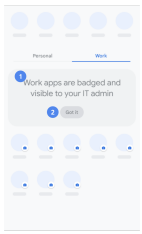
End new requirements

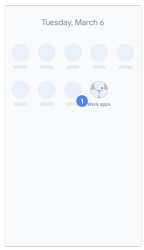
11.1.4 Work profile user education

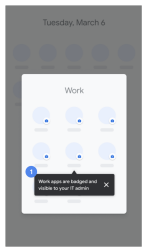
Start new requirements for 12

[GMS-11.1.4-001] In DEVICEs launching or upgrading to Android 12 or higher that report `android.software.managed_users`, the launcher MUST display an education message below the work tab or work folder after work profile provisioning is completed.

[GMS-11.1.4-002] In DEVICEs launching or upgrading to Android 11 that report `android.software.managed_users`, the launcher MUST display an education message below the work tab, work folder or work apps after work profile provisioning is completed. Additionally and optionally, the launcher may display an education message under the personal tab or personal apps.







[GMS-11.1.4-003] It's STRONGLY RECOMMENDED to use the AOSP string resource (and its localizations) as is, but if changed, the meaning of the changed string must be consistent with that of AOSP.

End new requirements

11.2 Managed device or profile apps installer policy

[GMS-11.2-001] Apps with `INSTALL_PACKAGES` permission that are enabled on a managed device or managed profile MUST only install apps required for the operation of the device or profile, apps specifically requested to be installed by the managing enterprise, or apps that fulfill an enterprise use case.

11.3 Setup experience

Start new requirements for 12

[GMS-11.3-001] DEVICEs MUST support fully managed device (device owner) and work profile provisioning in the Setup Wizard. In particular:

- **[GMS-11.3-001.001]** DEVICEs that report `android.hardware.nfc` MUST enable NFC bump-based provisioning. AOSP contains a reference implementation.
- **[GMS-11.3-001.002]** DEVICEs running Android with a camera MUST support the triggering of QR code-based provisioning through six taps on the Setup Wizard Welcome screen.
- **[GMS-11.3-001.003]** Any Setup Wizard customization MUST NOT disable or prevent fully managed device setup (device owner) and work profile setup methods supported in Google Mobile Services.

[GMS-11.3-002] In DEVICEs launching or upgrading to Android 12 or higher, if a work profile is set up (profile owner is established) during the Setup Wizard flow, then:

- **[GMS-11.3-002.001]** The user must be shown the Google Account screen in the primary profile (not in the work profile) to enable adding a personal Google account to the device.
- **[GMS-11.3-002.002]** If a personal account is added as described above, then the user must go through the setup flow they would see after adding a personal Google account, except for any steps which have already been completed.

End new requirements

11.4 Enterprise required apps

When a DEVICE is used as a fully managed device (Device Owner), or a Work Profile (Profile Owner) is created, a minimal set of apps required for operation of the device must be enabled:

- Automatically enabling any app without a launcher icon. This assumes such packages are vital system components.
- Using a set of XML allowlists to specify which apps with launcher icons are also vital to device operation in fully managed and Work Profile modes.

Required apps in a fully managed device

[GMS-11.4-001] In fully managed device mode, the following device functions are considered vital and MUST be added to the allowlist:

Start new requirements for 12

- **[GMS-11.4-001.001]** On DEVICEs running Android 11 or lower:
  - Google Play services (GmsCore)
  - Google Play Store
  - Google Setup Wizard
  - Default contacts app
  - Default downloads app
  - Default dialer
  - Default launcher
  - Default settings
- **[GMS-11.4-001.002]** On DEVICEs running Android 12 or higher:
  - Google Play services (GmsCore)
  - Google Play Store
  - Google Setup Wizard
  - Default contacts app
  - Default downloads app
  - Default dialer

- Default launcher
- Default settings
- Google app (Google Go app and Google Assistant Go for Go devices) if preloaded on the device.
- Android Auto stub (`com.google.android.projection.gearhead`) if preloaded on the device.

End new requirements

Required apps in a Work Profile

[GMS-11.4-002] In a Work Profile, the following device functions are considered vital and MUST be added to the list of required apps.

- **[GMS-11.4-002.001]** On DEVICEs running 10 or lower:
  - Google Play services (GmsCore)
  - Google Play store
  - Google Setup Wizard
  - Default contacts app
  - Default downloads app

Start new requirements for 12

- **[GMS-11.4-002.002]** On DEVICEs running Android 11:
  - Google Play services (GmsCore)
  - Google Play Store
  - Google Setup Wizard
  - Default contacts app
  - Default downloads app
  - Default wellbeing app
  - Google app (Google Go app and Google Assistant Go for Go devices) if preloaded on the device.

- **[GMS-11.4-002.003]** On DEVICEs running Android 12 or higher:
  - Google Play services (GmsCore)
  - Google Play Store
  - Google Setup Wizard
  - Default contacts app
  - Default downloads app
  - Default wellbeing app
  - Google app (Google Go app and Google Assistant Go for Go devices) if preloaded on the device.
  - Android Auto stub (`com.google.android.projection.gearhead`) if preloaded on the device.

End new requirements

Miscellaneous requirements

- **[GMS-11.4-003]** The default launcher and the default dialer SHOULD be enabled in a fully managed device or in ephemeral user (https://developer.android.com/work/dpc/dedicated-devices/multiple-users#overview) mode, but SHOULD NOT to be enabled in a Work Profile.
- **[GMS-11.4-004]** The contacts app MUST handle intents with the content type `vnd.android.cursor.dir/raw_contact`.
- **[GMS-11.4-005]** The downloads app MUST handle the intent action `android.intent.action.VIEW_DOWNLOADS`.
- **[GMS-11.4-006]** The intent action `android.settings.SETTINGS` MUST be handled in a fully managed device, Work Profile, or ephemeral user mode. In a fully managed device or ephemeral user mode, this MUST be handled directly by an app. In a Work Profile, this MAY be handled either by a cross-profile intent filter or by a settings app installed in the profile.
- **[GMS-11.4-007]** If the Google app is set as the default Assist app in the Personal Profile (primary user profile) (/gms/policies/domains/user-experience#assistant-default-assist-app), it MUST also be set as the default Assist app in the Work Profile.

11.5 Enterprise default cross-profile apps

Start new requirements for 12

~~**[GMS-11.5-001]** On DEVICEs running Android 11, to ensure appropriate out-of-box cross-profile user experiences, the following apps MUST be pregranted with the `INTERACT_ACROSS_PROFILES` permission:~~

- ~~Google app, if it's preloaded.~~
- ~~Gboard app, if it's preloaded and the out-of-box default IME app.~~

[GMS-11.5-002] On DEVICEs running Android 12 or higher, to ensure appropriate out-of-box cross-profile user experiences, the following apps MUST be pregranted with the `INTERACT_ACROSS_PROFILES` permission:

- Android Auto app, if it's preloaded.
- Google app, if it's preloaded.
- Gboard app, if it's preloaded and the out-of-box default IME app.

11.6 Cross-profile integration requirements

[GMS-11.6-001] In DEVICEs launching or upgrading to Android 12 or higher that report `android.software.managed_users` the following cross-profile integration must be supported.

Default Calendar

[GMS-11.6-002.001] The default Calendar app MUSTallow users to view personal calendar information in the work Calendar app using `INTERACT_ACROSS_PROFILES`.

[GMS-11.6-002.002] The default Calendar app MUST NOT store personal calendar information in the work profile, and MUST NOT send personal calendar information off the device, and MUST NOT make personal calendar information available to work apps like the DPC. Default IME app.

Default IME

[GMS-11.6-003] The default Input Method Editor MUST sync keyboard behavior settings across both profiles using `INTERACT_ACROSS_PROFILES` consent flow. Examples include:

- Vibration of keystrokes
- Keyboard layout
- Languages
- Personalization features MUST not be synced across profiles, including:
  - Predictive text or content
  - Voice typing data
  - Suggested contacts
  - Usage patterns
  - Learned words or dictionaries

End new requirements

12 Platform Policies

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Update in upcoming GMS requirements release

Release	Requirements	Links to requirements
March 2022	12.1	Location privacy policy (#location-privacy)

March 2022	12.5	Android Biometrics policy (#biometrics)

12.1 Location privacy policy

DEVICEs launching with Android 10 or higher MUST always:

- Respect the DEVICE location setting from Android Settings menu, not delivering any locations to apps or other software when this setting is Off.
- Display an ongoing indicator on the device screen and accurately attribute when a GPS location is accessed.

DEVICEs launching with Android 12 or higher and upgrading to Android 12 or higher MUST always:

- Respect app location permissions state (including background and coarse vs. fine location) when deciding whether and how to permit an app (or other software) to receive device location information. This requirement applies to all aspects and components of the DEVICE, including outside of the Android framework. This policy applies to the Android framework, apps, and software running on the main app processor, and the policy also applies to every hardware component, such as the modem, daemons, and any other submicrocontroller systems, and the software that may run thereof.

DEVICEs MAY ONLY access or send, or enable access to or use of, a device location outside of the Android framework when the DEVICE location setting is set to Off in the case of user-initiated emergency calls (such as 911, 112, or 123) to fulfill emergency service obligations solely to the extent required, and in the manner specifically directed, by law.

12.1.1 Location access

- To provide users with improved transparency of when their device location or activity is accessed, the device implementations:
  - MUST call `AppOpsManager#noteOp...()` to log each and every programmatic location access (including, but not limited to, the Android APIs such as `LocationManager#getCurrentLocation`and `LocationManager#getLastKnownLocation`) from Android device activities and services.

- MUST only use the list of `attributionTag` (https://developer.android.com/guide/topics/data/audit-access) instances declared by the location providers and current activity recognizers with the meta-data tag `android:location_allow_listed_tags` or `android:activity_recognition_allow_listed_tags` when logging location or activity recognition access events when:
  - It's from a location provider to derive further location information for the sole purpose of distributing location information to other on-device software components.
  - It's from an activity recognizer to recognize an activity for the sole purpose of reporting the activity information to other on-device software components.

Google's fused location provider, network Location provider and activity recognizer implementation in GMScore and the platform implementation in AOSP already meets the above requirements.

See Location Provider (/gms/policies/domains/location) for the integration guide.

12.2 System behavior

12.2.1 Execution of arbitrary commands

The operating system and preinstalled apps must not support executing arbitrary commands provided by other processes, for example, by passing commands to `Runtime.exec`, `ProcessBuilder`, or `system()`.

12.2.2 Signature permissions

There can't exist a mechanism to proxy or bypass the protection level of platform permissions, for example, providing an API or interface.

12.2.3 Platform signing policy

The platform key MUST ONLY be used to sign applications for which the device manufacturer (MADA signatory) has all of the source code or other evidence, approved by Google, of security audits on the code. The device manufacturer MUST have the capability to fix and remediate any security issue identified in the application signed with the platform

key. Beginning in 2020, performing a security audit is a prerequisite for signing with the platform key.

Nonpreloaded apps signed with platform signatures pose a high security risk because they give system-level privileges to nonsystem apps. Sharing platform certificates with multiple DEVICEs (/gms/policies/domains/introduction#definitions) also creates a single point of failure. Hence as a new policy:

- It's STRONGLY RECOMMENDED that ONLY preloaded apps can be signed with a platform signing certificate. Such apps should be a full version of the app (not a stub (/gms/policies/domains/introduction#definitions)).
- It's also STRONGLY RECOMMENDED that OEMs use different platform certificates for their different DEVICE (/gms/policies/domains/introduction#definitions) builds.

12.3 Emergency location bypass policy

12.3.1 Behavior of Emergency Location Bypass API

- If any additional location restrictions (for example, a new type of location throttling) are added, the API MUST be able to bypass such restrictions as with all other restrictions.
- DEVICEs running Android 10 or higher MUST ensure that only apps in the allowlist can access the API.
- Partners MAY add apps to the allowlist, but allowlisted apps MUST support user-initiated emergency sessions to a Public Safety Answering Point (PSAP).
- GMS Core MUST be added to the allowlist out-of-box for Android Emergency Location Service.
- Partners MUST NOT add additional security restrictions to the API that would prevent GMS Core from using the API.

12.4 Custom system permissions, UIDs, and APIs policy

DEVICEs (/gms/policies/domains/introduction#definitions) launching with Android 12 or higher and submitted on the APA portal on or after Sep 8, 2021, partners MUST fix custom permissions, UIDs, and APIs policy violations on the software build

(/gms/policies/domains/introduction#definitions) and every MBA (/gms/policies/domains/introduction#definitions) in the software build.

Additionality, the following types of MBAs using any custom system permission, UID, or API that violates this policy MUST NOT be loaded on PRODUCTs (/gms/policies/domains/introduction#definitions) running Android 9 or higher on or after Sep 8, 2021:

- Any newly added MBA (/gms/policies/domains/introduction#definitions) in the build (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))
- Existing MBA (/gms/policies/domains/introduction#definitions) (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) that wasn't using a new custom system permission, UID, or API earlier but started using any new custom permission, UID, or API after Sep 8, 2021 is considered as a newly added MBA.

The custom system permissions, UIDs, and APIs MUST meet the following criteria:

- User data (/gms/policies/domains/mba#mba-user-data) access that's protected by AOSP permissions MUST ONLY be protected by AOSP permissions and there MUST NOT be alternate means (for example, custom system permissions, APIs, UIDs) to get the same data.
- User data (/gms/policies/domains/mba#mba-user-data) MUST NOT be exposed through alternative means that have a similar or weaker protection mechanism than what's provided by AOSP.
- There MUST NOT be a mechanism to proxy or bypass the protection level of platform permissions, for example, providing an API, permission, UID, or interface.
- Platform APIs gated by system permissions MUST NOT be duplicated by custom APIs.
- Apps running under custom system UIDs MUST NOT access user data (/gms/policies/domains/mba#mba-user-data) for which there are existing MBA policies (/gms/policies/domains/mba) and Platform policies (/gms/policies/domains/platform-policies) (for example, Android APIs that are protected by dangerous permissions (https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions)).

12.5 Android Biometrics policy

DEVICEs launched with Android 10 or higher with the applicable biometric sensor class MUST report device biometrics test results in a Biometric Compliance Report (BCR) as shown in the table below.

Initial Release (IR)	**BCR reporting requirements**	**SAR testing requirements**	**Lab requirements**[1] (#security-lab)	**Test protocol requirements**[2] (#test-protocols)
12 (S)	Required	**Class 3** or **Class 2**: required **Class 1**: strongly recommended	**Class 3** or **Class 2**: lab required Others: Can self-attest	Required: Android 12 protocol
11 (R)	Required	**Class 3** or **Class 2**: required **Class 1**: strongly recommended	**Face 2D Class 3** or **Class 2**: lab required Others: Can self-attest	Required: Android 11 protocol for **Face 2D Class 3** or **Class 2**, and Android 10 protocol for other modalities. Strongly recommended: Android 11 protocol
10 (Q)	**Class 3** or **Class 2**: required Class 1: strongly recommended	**Class 3** or **Class 2**: required **Class 1**: strongly recommended	**Face 2D Class 3**: Must use a third-party lab **Face 2D Class 2**: Waiver granted (can self-attest) Others: Can self-attest	Required: Android 10 protocol Strongly recommended: Android 11 protocol
9 (P) or lower	Strongly recommended	Strongly recommended	Can self-attest	Required: None Strongly recommended: Android 10 protocol

1. For devices running Android 11 or higher, you must use a biometric security lab that meets Android requirements. Refer to the List of Testing Labs

(https://docs.partner.android.com/partners/guides/biometrics/labs) for more details.

2. The detailed test protocols of different Android OS versions can be in the Biometric Security Program (/gms/testing/overview/biometric).

Note: See Launch Approval (/gms/testing/overview#mandatory) page for BCR requirement for each software release type to clarify the requirement for each release type (IR, MR, SMR, and so on).

End new requirements

13 MBA Policies

Version 9.1, last updated March 14, 2022

See the following resources for other versions of the GMS Requirements.

- Preview GMS Requirements (/gms/policies/preview)
- Past GMS Requirements (/gms/resources/reqs).

Update in upcoming GMS requirements release

Release	Requirements	Links to requirements
March, 2022	13.2.5	• 13.2.5.1 Eligible device lock scenarios (#eligible-device-lock-scenarios) • 13.2.5.2 Device lock-down requirements for MBAs (#mba-locking-down-devices) • 13.2.5.4 User notifications before device lock (applicable to financed devices and subsidy devices) (#user-notification-device-lock)

March, 2022	13.3.1	Minimum target SDK (#min-target-sdk)

Introduction

Mobile bundled apps (MBAs) are:

- apps that a device manufacturer or mobile network operator bundles by preloading in the shipping software flashed onto the device.

or

- apps that are distributed over the air and installed on a user's device at any time (and not just during or shortly after the setup phase) by the device manufacturer's services or other preloaded third-party installers, without the user explicitly choosing to download and install them.

Even when the user is presented with an option to install an app, the app is considered an MBA if the user's choice isn't explicit. The following are examples of when apps are considered MBAs:

- The user is presented with a list of apps to be installed but with checkboxes selected by default.
- The user is presented with a notification to install an app but with a single-click to install, without an alternative option presented equally.

Any MBA that is also updated through Google Play (that's, an app on Google Play with a matching package name and the same signature) MUST adhere to the Google Play developer policy (https://play.google.com/about/developer-content-policy/#!?modal_active=none).

13.1 Content policy

For every software build for GMS devices submitted on Android Partner Approvals (APA) on or after:

- **August 1, 2019**: MUST adhere to this content policy.

This includes:

- Every MBA in the BUILD (if it's for a new PRODUCT)

- Any newly added MBA in the build (if it is for a previously approved PRODUCT)

13.1.1 Restricted content

MBAs MUST NOT include restricted content.

Per our commercial agreement terms, below are some examples of when an MBA is considered as engaging in an activity that violates an applicable law or regulation (and thus is considered restricted content):

- **Child endangerment:** Apps that include content that sexualizes minors.
- **Illegal or prohibited activities:** Apps that facilitate or promote illegal activities such as the sale or purchase of illegal drugs, or encourage the illegal use or production of drugs, prohibited weapons, self-harm, nonconsensual sex, or other illegal sexual themes.

Further, below are examples of categories of restricted content. For more examples of restricted content, see the MBA Content Policy Supplement (/gms/policies/overview/mba-content-policy).

- **Inappropriate content**
    - **Sexual content:** Apps that contain or promote sexual content, such as pornography, nudity or suggestive illustrations in an erotic context, or promoting sexual acts in exchange for compensation. App content that contains nudity may be allowed if the primary purpose is educational, documentary, scientific, or artistic, and isn't gratuitous.
    - **Hate speech:** Apps that contain, promote, glorify, condone, or incite hatred, discrimination, bullying, or violence against any individual or group.
    - **Violence:** Apps that contain graphic depictions of realistic, gratuitous violence or violent threats to any person or animal, and content related to terrorism, such as content that promotes terrorist acts, incites violence, or celebrates terrorist attacks or hate groups. Apps that promote self harm, suicide, eating disorders, choking games, or other acts where serious injury or death may result.
    - **Sensitive events:** Apps that promote insensitivity toward or capitalize on a natural disaster, atrocity, conflict, death, or other tragic event.
    - **Bullying and harassment:** Apps that contain or facilitate threats, harassment, or bullying.

- **Marijuana:** Apps that facilitate the sale of marijuana or marijuana products, regardless of legality.
- **Tobacco and alcohol:** Apps that facilitate the sale of tobacco (including e-cigarettes) or encourage the irresponsible use of alcohol or tobacco.

- **Financial products and services**
  - **Binary options and cryptocurrencies:** Apps that provide users with the ability to trade binary options and contain real-money trading functionality. Apps that perform cryptocurrency mining on DEVICEs. However, apps that remotely manage the mining of cryptocurrency are permitted to be preloaded on DEVICEs.
  - **Personal loans:** We define personal loans as lending money from one individual, organization, or entity to an individual consumer on a nonrecurring basis, not for the purpose of financing the purchase of a fixed asset or education. Personal loan consumers should be provided with information about the quality, features, fees, risks, and benefits of loan products in order to make informed decisions about whether to undertake the loan.

    Examples: Personal loans, payday loans, peer-to-peer loans, or title loans. Not included: Mortgages, car loans, student loans, or revolving lines of credit (such as credit cards, or personal lines of credit).

    Apps for personal loans MUST disclose the following information in the app metadata:

    - Minimum and maximum period for repayment
    - Maximum annual percentage rate (APR), which generally includes interest rate plus fees and other costs for a year, or similar other rate calculated consistently with local law
    - A representative example of the total cost of the loan, including all applicable fees

    We don't allow apps that promote personal loans that require repayment in full in 60 days or less from the date the loan is issued. This policy applies to apps that offer loans directly to consumers, lead generators, and those who connect consumers with third-party lenders.

    High APR personal loans: In the United States, we don't allow apps for personal loans where the APR is 36% or higher. Apps for personal loans in the United States MUST display their maximum APR, calculated consistently with the Truth in Lending Act. This policy applies to apps that offer loans directly to

consumers, lead generators, and those who connect consumers with third-party lenders in the United States.

- **Gambling:** Preloading of gambling apps is restricted in geographies where gambling is prohibited or the developer doesn't have a valid license to operate online gambling apps. The definition of *gambling* is determined per the applicable laws.
- **Unapproved substances:** Apps that promote or sell unapproved substances, irrespective of any claims of legality.
- **User-generated content (UGC):** Content generated by users is considered a part of the app for content policy purposes. UGC is content that users contribute to an app, and that's visible to or accessible by at least a subset of the app's users. Objectionable content is content that violates our policies. We prohibit apps whose primary purpose is (1) featuring objectionable UGC, (2) hosting objectionable UGC, or (3) developing a reputation among users of being a place where such content thrives. MBAs that contain or feature UGC MUST:
  - Implement a robust, effective, and ongoing UGC moderation to remove restricted and prohibited content and take enforcement actions on offenders to prevent them from being flagged against this requirement.
  - Provide safeguards to prevent in-app monetization from encouraging objectionable user behavior.

13.1.2 Impersonation and intellectual property

Per our commercial agreement terms, the use of Google trademarks is subject to the Google Mobile Branding Guidelines (https://partnermarketinghub.withgoogle.com/#/). MBAs MUST NOT infringe, induce, or encourage the infringement of the intellectual property rights of Google (including trademark, copyright, patent, trade secret, and other proprietary rights). This includes, but is not limited to:

- Apps that impersonate Google or imply a relationship or affiliation with Google where none exists.
- Apps that induce infringement of Google products or services, such as YouTube downloader, when such functionality doesn't exist or isn't approved.
- App titles and icons that are so similar to those of existing Google products or services that users might be misled.

Additionally, Google reserves the right to reject build approvals when there is an MBA which Google is made aware of to be impersonating others without authorization, or infringe on

the intellectual property rights of other content owners.

DEVICE manufacturers are responsible for implementing control mechanisms to enforce against violation of intellectual property rights of other content owners, including measures such as responding to clear notices of alleged copyright infringement.

13.1.3 Deceptive or disruptive ads

MBAs that serve ads without meeting the following guidelines are considered deceptive or disruptive and aren't allowed:

Deceptive ads:

- **MUST NOT** simulate or impersonate the user interface of any app, notification, or warning elements of an operating system.
- **MUST** be clearly attributed to the source (for example, the originating app) for the end users.

Disruptive ads:

- **MUST NOT** be shown in a way that results in inadvertent clicks. It's prohibited to force users to click an ad or submit personal information for advertising purposes before users can fully use the app.
- **MUST NOT** force users to interact with it (for example, click-through on the ad, watch the video ad for minimum X seconds) to access the core functionality of the DEVICE that isn't part of the app itself, including, but not limited to, full-screen ads or lock-screen ads that aren't dismissible.

13.1.4 Deceptive behavior

MBAs that attempt to deceive users or enable dishonest behavior aren't allowed. MBAs MUST NOT interfere with an end user's expected operation and use of the DEVICE. These include, but aren't limited to, the following prohibited deceptive behaviors:

- **Misleading claims:** MBAs whose app name or icon misleads the users about app functionality or MBAs that make false or misleading claims about other apps or services.
- **Imitation of system functionality:** MBAs (which aren't part of the DEVICE system) that mimic or interfere with system functionality (such as notifications, warnings, or user

confirmation dialogs) to mislead users into thinking that the behavior is coming from the OS itself instead of the app.

- **Deceptive device settings changes:** MBAs (which aren't part of the DEVICE system) that make deceptive changes to the user's DEVICE settings or features outside of the app without the user's knowledge and explicit consent.
- **Enabling dishonest behavior:** MBAs that help users mislead others, including, but not limited to, apps that enable or facilitate dishonest behavior, such as generating fake ID documents (for example, passports, driver's licenses, diplomas, credit cards, or social security numbers).
- **Spam:** MBAs that spam users, such as apps that send users unsolicited messages and lack clear in-app controls to stop such behavior. Additionally, MBAs whose primary purpose is to drive attributed downloads of another app, drive affiliate traffic to a website, or provide a webview of a website aren't allowed.
- **Lacking minimum functionality:** MBAs that have an unstable or unresponsive user experience, such as apps that crash, force close, freeze, or otherwise function abnormally while performing the core services.

13.2 MBA privacy policies

13.2.1 User data policy

For new launching DEVICEs submitted through Android Partner Approvals (APA) portal on or after January 4, 2021, partners MUST fix data collection violations within 60 days of the date of the disclosure. These fixes MUST be applied to every MBA in the software build.

Apps must be transparent in how they handle user data (for example, information collected from or about a user, including device information). That means disclosing the access, collection, use, and sharing of the data, and limiting the use of the data to the purposes disclosed. In addition, if apps handle personal or sensitive user data, the additional requirements in Personal and sensitive information (#personal-sensitive-info) apply in addition to any requirements prescribed by applicable privacy and data protection laws.

13.2.1.1 Personal and sensitive information

Personal and sensitive user data includes, but isn't limited to, personally identifiable information, financial and payment information, authentication information, phonebook, contacts, device location (https://developer.android.com/training/location), SMS and call-related

data, microphone, camera, app accessing a user's inventory of installed apps, and other sensitive device or usage data.

If the app handles personal or sensitive user data, then the app:

- MUST limit the access, collection, use, and sharing of personal or sensitive data acquired through the app to purposes directly related to providing and improving the features of the app (for example, user-anticipated functionality that's documented in the app's description before it's downloaded on the app distribution platform, or presented during the out-of-box setup flow for preloaded apps, or by being the default handler for certain core Android functionalities (https://android.googlesource.com/platform/frameworks/base/+/master/services/core/java/com/android/server/pm/permission/DefaultPermissionGrantPolicy.java)).
- MUST provide a privacy policy to the user that comprehensively discloses how the app accesses, collects, uses, and shares user data. The app's privacy policy must disclose the type of parties to which any personal or sensitive user data is shared. In other words, for user-downloaded apps, post a privacy policy within the app itself as well as in the app distribution platform's console; and for manufacturer-distributed apps, post a privacy policy within the app, or when it's a manufacturer-distributed headless app in the settings or out-of-box setup flow.
- MUST obtain the access to data protected by dangerous and special Android permissions (https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions) using one of these approaches:
    - Requesting the permission at runtime through the permission request guidance (https://developer.android.com/guide/topics/permissions/overview#dangerous-permission-prompt).
    - Having the permission granted by the Android system through the Pregrant permissions policy (#mba-pregrant-permissions).
        - MUST use the data only for purposes that the user has agreed to (subject to exemptions under the Pregrant Permissions Policy (#mba-pregrant-permissions). If apps later need to use the data for other purposes, apps must ask users and make sure they agree to the additional uses.
- MUST handle all personal or sensitive user data securely, including transmitting it using modern cryptography (for example, over HTTPS).

- All other transfers or sales of the user data are prohibited, unless a transfer is necessary to comply with applicable laws or as part of a merger, acquisition, or sale of assets with legally adequate notice to users.

13.2.1.2 Prominent disclosure and consent requirement

In cases where users may not reasonably expect that their personal or sensitive user data is required to provide or improve the policy-compliant features or functionality within the app (for example, data collection occurs in the background of the app), apps MUST provide transparency to users on data collection and use by meeting the following disclosure and consent requirements.

The disclosure

- MUST be prominently presented (preferably within the normal usage of the app itself, that is, either without the need to navigate into a menu or settings, or if the app isn't typically opened by the user then during the out-of-box setup flow or notifications with clear attributions to the app) to the user through the device interface (for example, on screen) in addition to a disclosure in the privacy policy.
- MUST be presented as an independent disclosure and not be included with other disclosures unrelated to user data.
- MUST include a comprehensive list of the types of data (for example, SMS, contacts, location) being accessed or collected.
- MUST explain the purposes for which the data will be accessed, used (for example, which features and functionality the data will support or how the data will be used to improve the app), collected (for example, where the data will be stored), and shared with other entities (third party or affiliated).

The user consent interface

- MUST accompany and immediately follow the disclosure.
- MUST be presented before the access or collection of any personal or sensitive data.
- MUST present the consent dialog clearly and unambiguously.
- MUST require an affirmative user action (for example, tap to accept, select a checkbox) in order to accept.
- MUST NOT interpret navigation away from the disclosure (including tapping away or pressing the back or home button) as consent.

- MUST be presented until there's an affirmative action and not use autodismissing or expiring messages (for example, toasts (https://developer.android.com/guide/topics/ui/notifiers/toasts)).

Here are some examples of violations

- An app that accesses a user's inventory of installed apps and doesn't treat this data as personal or sensitive data subject to the privacy policy, data handling, and prominent disclosure requirements.
- An app that accesses a user's phone or contact book data and doesn't treat this data as personal or sensitive data subject to the privacy policy, data handling, and prominent disclosure requirements.
- An app that records a user's screen and doesn't treat this data as personal or sensitive data subject to this policy.
- An app that collects device location (https://developer.android.com/training/location) and doesn't comprehensively disclose its use in accordance with the above requirements.
- An app that collects restricted permissions in the background of the app including for tracking, research, or marketing purposes and doesn't comprehensively disclose its use and obtain consent in accordance with the above requirements.

13.2.1.3 Restrictions for sensitive data access

In addition to the requirements above, the table below describes requirements for specific activities.

Activity	Requirement
App handles financial or payment information or government identification numbers	App must never publicly disclose any personal or sensitive user data related to financial or payment activities or any government identification numbers.
App handles nonpublic phonebook or contact information	App must never publish or disclose nonpublic contacts without user's authorization.

See the MBA user data policy (/gms/policies/overview/user-data) for additional details.

13.2.2 Pregrant permissions policy

The following definitions are based on the announcement (August 1, 2019) and enforcement (February 3, 2020) dates of this policy.

- **Existing app:** All requested pregrants were approved for the app prior to the announcement date. An existing app requesting additional pregrants is classified as a **new app**.
- **New app:** Requested pregrants weren't approved for the app prior to the announcement date.
- **Device approved before enforcement date:** APA approval happens before enforcement date.
- **Device approved after enforcement date:** APA approval happens after the enforcement date.

The table below defines the required compliance timeframe for the app/device permutation.

	Existing app	New app
Device approved before enforcement date	Exempt (to ensure no impact to UX and encourage updates)	Must comply in next software update (OS, MR, or SMR)
Device approved after enforcement date	Must comply by enforcement date	Must comply before device approval

All Android apps, including MBAs, MUST include user prompts for dangerous permissions (https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions) under Android's runtime permissions model. Pre-grants may be approved in the following cases.

	Case	Examples of use cases allowed exemptions	Examples of use cases not allowed exemptions
1	APK is required for users to set up the DEVICE.	Carrier account Setup Wizard, OEM/Carrier app	Carrier account Setup Wizard with location-

		store	based offers, eReader
2	APK is required for core DEVICE functionality that doesn't have an app-based UI.	OTA management, network quality management	Tracking store location where the DEVICE was sold, file manager
3	APK is first used in a scenario when a permission request on first run could have life-threatening or serious material consequences.	Emergency service notifications, DEVICE finder	Parental control, remote locking DEVICE for unpaid carrier bills

When runtime permissions are pre-granted in the cases above, an alternative UI (#alt-UI-screen) MUST be presented to the user. This screen MUST be shown before the app accesses any data guarded by the permission.

Note: The only cases exempt from presenting an alternative UI to the user are:

1. Carrier uses of `PHONE`, `CALL_LOG`, and `SMS` permissions, where this is covered in a Terms of Service (ToS).
2. Cases falling under exemption 3 above, when not being able to transmit the device's location puts the user in danger. For example, in the US, user-initiated E911 use cases do NOT require an alternative UI, but the use of the phone's location to target wireless emergency alerts to the user's device does require an alternative UI.
3. App store functionality.
4. For enterprise flows, aborting activation is considered sufficient user control/alternative UI.

Default intent handlers are automatically granted the set of permissions defined by AOSP (https://android.googlesource.com/platform/frameworks/base/+/master/services/core/java/com/android/server/pm/permission/DefaultPermissionGrantPolicy.java), and don't need to apply for pregrants for those permissions or supply an alternative UI. Some examples are in the table.

Intent handler	Permissions
Camera	Camera, microphone, storage

Dialer	Contacts, phone
SMS handler	Contacts, SMS

Where a default intent handler has been added to a version of AOSP, permissions granted to that default intent handler may be pregranted as exemptions under this policy to apps preloaded on BUILDs of previous versions of the platform.

Where the GMS Requirements mandate an APK to be set as a default intent handler across all DEVICEs, OEM or carrier APKs fulfilling the same use case may be pregranted the same permissions as that default handler. As with default handlers, an alternate UI isn't required. For example, alternatives to Google Assistant may be granted the permissions available to the default voice interaction handler.

Exemptions under this policy are granted for individual permissions and not at a permission group level.

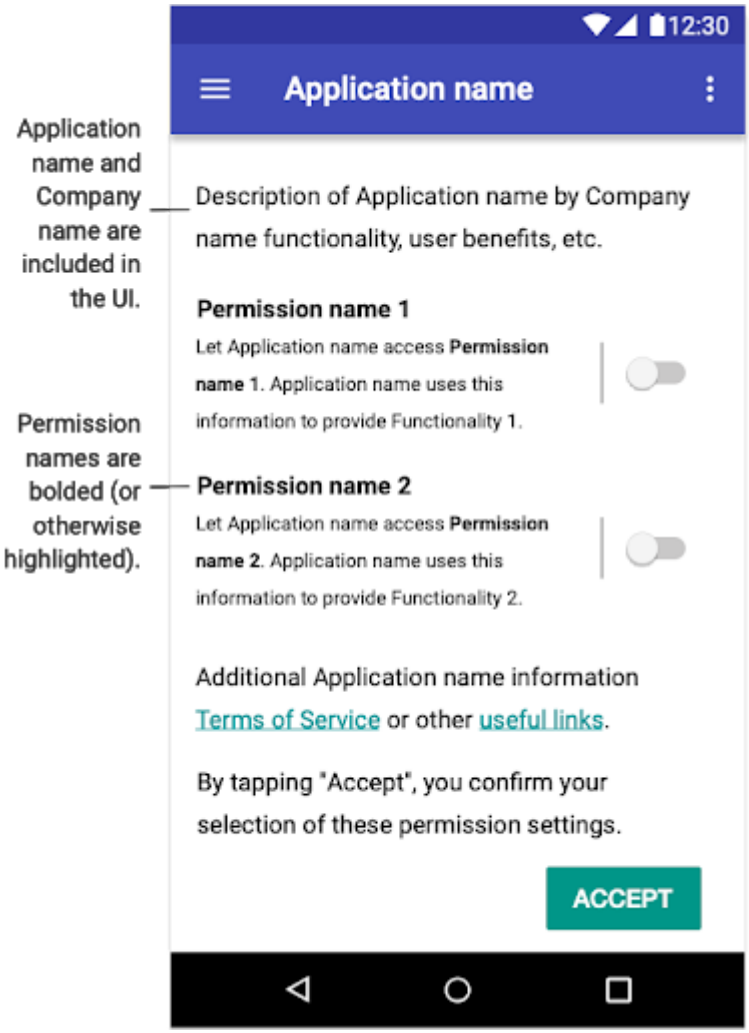
Any processing of user/DEVICE data MUST meet the Unwanted Software Policy (https://www.android.com/mobile-unwanted-software-policy) requirements.

13.2.2.1 Alternative UI

Apps requiring an alternative UI screen MUST comply with these requirements:

- Data obtained through the permission MUST not be accessed until the user has explicitly accepted through the alternative UI.
- The alternative UI screen MUST contain the following:
  - Name of the company (OEM or Carrier) providing the app
  - Name of the app requesting the permissions
  - Permissions being requested (bolded or otherwise highlighted in the body of text)
  - An individual toggle for each requested permission
  - An explicit affordance for the user to accept a set of permission decisions. A **Cancel**, **Later**, or **Next** button isn't permitted. It's also RECOMMENDED that if all permissions on the alternative UI are declined by the user, the text of the button should change to **Next**.

- The default setting of the individual permission toggles may be set to enabled or disabled.
- If any personal and sensitive information (#personal-sensitive-info) obtained through any of the requested pregrant permissions will be sent off the device, the alternative UI MUST also explain how the data will be used in order to be compliant with the MBA user data policy (#mba-user-data).



Example: Alternative UI

13.2.3 SMS and call log permissions policy

For new launching DEVICEs submitted through Android Partner Approvals (APA) portal on or after

- **January 4, 2021**: partners MUST fix SMS and call log policy violations to every MBA in the software build.

SMS and call log permissions are regarded as personal and sensitive user data subject to the personal and sensitive information (#personal-sensitive-info) policy, and the following

restrictions.

Restricted permission	Requirement
Call log permission group (for example, `READ_CALL_LOG`, `WRITE_CALL_LOG`, `PROCESS_OUTGOING_CALLS`)	The app must be actively registered as the default phone or Assistant handler on the device.
SMS permission group (for example, `READ_SMS`, `SEND_SMS`, `WRITE_SMS`, `RECEIVE_SMS`, `RECEIVE_WAP_PUSH`, `RECEIVE_MMS`)	The app must be actively registered as the default SMS or Assistant handler on the device.

Apps lacking default SMS, phone, or Assistant handler capability may not declare the use of the above permissions in the manifest. This includes placeholder text in the manifest. Additionally, apps must be actively registered as the default SMS, phone, or Assistant handler before prompting users to accept any of the above permissions and must immediately stop using the permission when they're no longer the default handler. The permitted uses and exemptions are available in SMS/call log permission groups (/gms/policies/overview/permission-groups).

Apps may use the permission (and any data derived from the permission) only to provide approved core app functionality. Core functionality is defined as the main purpose of the app. This may include a set of core features, which must all be prominently documented in the app's description on the app distribution platform, or presented during the out-of-box setup flow for preloaded apps, or by being the default handler for certain core Android functionalities (https://android.googlesource.com/platform/frameworks/base/+/master/services/core/java/com/android/server/pm/permission/DefaultPermissionGrantPolicy.java). Without the core features, the app is broken or rendered unusable. The transfer, sharing, or licensed use of this data must be only for providing core features or services within the app, and its use may not be extended for any other purpose (for example, improving other apps or services, advertising, or marketing purposes). Apps may not use alternative methods (including other permissions, APIs, System UIDs or third-party sources) to derive data attributed to call log or SMS-related permissions.

See SMS/Call Log Permissions (/gms/policies/overview/sms-call-log) for additional details.

Permission requests

Permission requests should make sense to users. Apps may request only the permissions that are necessary to implement current features or services in the app. Apps may not use permissions that give access to user or device data for undisclosed or disallowed features or purposes.

Apps should request permissions to access data in context (through incremental auth), so that users understand the reason for the request.

Respect users' decisions if they decline a request for a restricted permission, in other words:

- Users may not be manipulated.
- Users may not be forced into consenting to any noncritical permission.
- Apps must provide alternative methods to accommodate users who don't grant access to sensitive permissions (for example, allowing a user to manually enter a phone number if they've restricted access to call logs).

Certain Restricted Permissions may only be requested in rare cases where apps provide a highly compelling or critical feature, and where there is no alternative method available to provide the feature.

13.2.4 MBA personal profile data privacy policy

13.2.4.1 Personal profile data privacy requirements

DEVICE implementations that support the work profile (`android.software.managed_users`):

- MUST NOT preload or distribute software components out-of-box that share or make visible personal profile data to apps within the work profile.
- MUST NOT preload or distribute software components out-of-box that have, or allow other apps running in the work profile to enforce device policy controls (https://developer.android.com/work/dpc/device-management) in the personal profile, unless the device is provisioned as a company-owned device in accordance with the AOSP implementation of device ownership.

For all work profile scenarios, including when the device is provisioned as a **company-owned device**, software components preloaded or distributed out-of-box:

- MUST NOT send personalization data or any other type of personal profile data to the work profile for the purposes of aggregating app usage statistics or user behavior across the device.

- MUST NOT reveal personal profile data by either direct or indirect means, for example, accepting a package name as an API parameter that returns `true` only if that package is present in the personal profile.
- Ensure that any personal profile data shared with apps or software running in the work profile MUST NOT be sent off the device by apps or software running in the work profile.

13.2.4.2 Cross-profile communication privacy requirements

On DEVICEs running Android 11 or higher that support managed profiles,

- Platform-signed apps can declare and use the `INTERACT_ACROSS_PROFILES` permission if they're one of the AOSP components such as `com.android.shell`.
- Preloaded Assist apps or preloaded out-of-the-box default Input Method Editor (IME) app may be pregranted with the `INTERACT_ACROSS_PROFILES` permission without user consent, subject to the Cross profile prergrant restrictions (#cross-profile-pregrant-restrict) defined below.
- All other apps MUST get user consent by displaying appropriate consent screens in accordance with the AOSP implementation before the `INTERACT_ACROSS_PROFILES` permission is granted.

Note: An Assist app must implement a VoiceInteractionService or an `ACTION_ASSIST` intent handler.

13.2.4.3 Cross-profile pregrant restrictions

When the `INTERACT_ACROSS_PROFILES` permission is pregranted, these apps:

- MUST NOT send Personal Profile data to the Work Profile; any features that send Personal Profile data to the Work Profile MUST be optional for use of the device, and provide additional user-facing consent similar to that implemented in AOSP.
- MAY send data unrelated to the Personal Profile data such as static settings, preferences, or configuration data that do not relate to users' personal data or user-generated content, from the Personal Profile to the Work Profile for the purposes of device configuration.
- MUST comply with the other Personal Profile data privacy requirements (#personal-profile-data-privacy-req) in all cases.

13.2.5 MBA Device Lock policy

Software builds for new DEVICEs (https://docs.partner.android.com/gms/policies/domains/introduction#definitions) launching with Android 10 or higher submitted on the APA (Android Partner Approvals) portal on or after:

- **September 5, 2021** will receive `WARNING` message if preloading a non-compliant app.
- **January 3, 2022** MUST comply with the MBA Device lock policy and fix any violations on every MBA (https://docs.partner.android.com/gms/policies/domains/introduction#definitions) in the software build (https://docs.partner.android.com/gms/policies/domains/introduction#definitions).

Code that meets any of the following is conventionally classified by Google as Ransomware (https://support.google.com/googleplay/android-developer/answer/9888380#ransomware) and is not allowed distribution.

- Takes partial or full control over a device or its data without a user-initiated action.
- Demands payment or action other than removing the SIM from a different network provider to release control.

MBAs may implement device-locking solutions to manage devices if they meet the following MBA Device Lock policy requirements.

Note: For more information, see MBA Device Lock Policy enforcement (/gms/policies/domains/device-lock-enforcement).

13.2.5.1 Eligible device lock scenarios

- Devices distributed with an agreement to be paid-in-full by the user over a certain pre-defined term and locks the device on missed payments (referred to as Financed Devices).
- Devices that are pre-locked from the factory before reaching the user.
- Locking solutions implemented as an app or Android framework component that can lock the device functionality based on SIM insertion and/or removal.
- Devices sold under a rebate or discount plan with an agreement that the user will remain with the service plan for a certain pre-defined term and locks the device on

missed service payments, recharge, or removal of a valid SIM (referred to as subsidy devices).

13.2.5.2 Device lock-down requirements for MBAs

The device locking solution:

- MUST be preloaded or installed during the Setup Wizard flow before the out-of-box setup flow is completed.
- MUST be, after being paid-in-full by the user or after fulfilling any contractual obligations, immediately disabled on the devices and MUST NOT be re-enabled even after the device is factory reset. For solutions where the MBA client supports other functionality, they MAY alternatively disable the device lock functionality on the server. For devices that are SIM-locked, the user MUST be provided with a path for disabling the lock on the device.
- MUST encrypt and authenticate all remote lock/management commands.
- MUST NOT exploit security vulnerabilities, non-documented behaviors of public APIs, or hidden APIs as a means to implement the functionality (as these may get "patched" in future security updates).
- For financed devices and subsidy devices that lock the device on missed payments, see the additional requirements below.
    - MUST NOT block the out-of-box setup flow unless the device identifier maps to a device not paid-in-full by the user.
    - MUST show a prominent disclosure (#device-lock-disclosure) accompanied with explicit user consent (#device-lock-consent) before actively monitoring the payment status of the device and blocking the device functionalities.
    - MUST afford the user a prominent and persistent warning, with minimum time in advance (#user-notification-device-lock) before full lock-down event for financed devices.
    - MUST provide a minimum one day warning period before the device is locked due to a missed payment or recharge when the device was in an offline state.
    - MUST provide the ability for the user to get a minimum of one day extension if the device is locked due to a missed payment or recharge when the device was in a powered-off state.
- SIM lock solutions:

- MUST NOT lock the device when no SIM is inserted as long as there are no missing payments for the device.
- MUST NOT lock the device when the SIM is from an authorized network provider as long as there are no missing payments for the device.

13.2.5.3 Prominent disclosure and consent requirement

MBAs distributed with the device whose primary purpose is to manage the device locking function for a device may be excluded from the ransomware category provided they successfully meet requirements for secure lock and management, and adequate user disclosure and consent requirements as detailed below. In addition, adequate user disclosure and consent requirements are required each time the account owner changes until the device is paid in full.

13.2.5.3.1 The disclosure

- MUST be presented without the need for the user to navigate into a menu or settings.
- MUST NOT be placed in a lengthy, off-device privacy policy or Terms of Service (ToS).
- MUST include a request for user consent (#device-lock-consent).

13.2.5.3.2 Explicit user consent

- MUST accompany and immediately follow the disclosure.
- MUST present the consent dialog clearly and unambiguously.
- MUST require an affirmative user action (for example, tap to accept, select a checkbox, and so on) to accept.
- MUST NOT interpret navigation away from the disclosure (including tapping away or pressing the Back or Home button) as a consent.
- MUST be presented until there's an affirmative action and not use auto dismissing or expiring messages.

13.2.5.4 User notifications before device lock (applicable to financed devices and subsidy devices)

The device users must be given a warning period in which to take action before the device is locked. See the table below for the minimum warning periods.

Payment plan	Minimum warning period before device is locked
Monthly	7 days
Weekly	5 days
Daily	1 day

For more information on Android's solution to enable device locking functionality, see Device Lock (/gms/building/integrating/device-lock).

For more information about the enforcement of this policy, see Device lock enforcement (/gms/policies/domains/device-lock-enforcement).

13.2.6 Notification access and notification listeners policy

For new launching DEVICEs submitted through the Android Partner Approvals (APA) portal on or after:

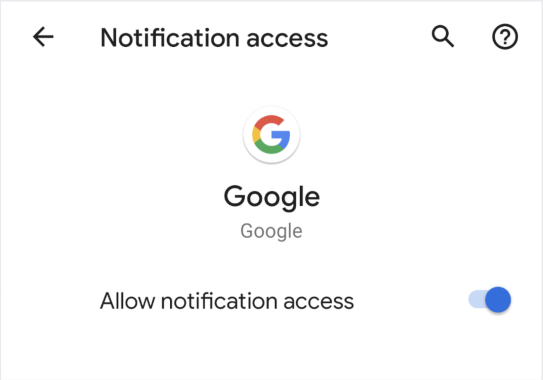
- **June 10, 2021**: partners MUST fix notification access and notification listeners policy violations for every MBA in the software build (/gms/policies/domains/introduction#definitions).

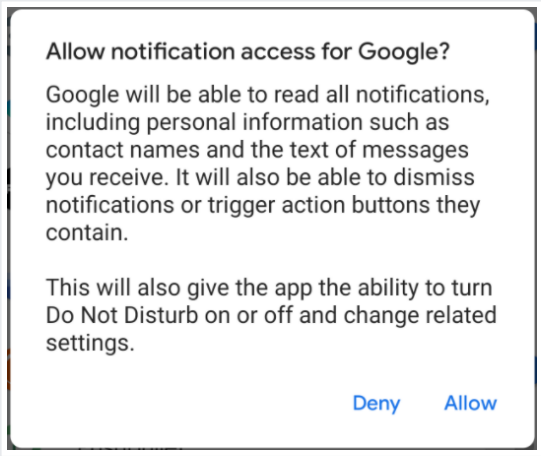
This includes:

- Every MBA (/gms/policies/domains/introduction#definitions) in the BUILD (if it's for a new launching DEVICE (/gms/policies/domains/introduction#definitions))
- Any newly added MBA (/gms/policies/domains/introduction#definitions) in the build (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))
- Existing MBA (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) adding any new permission/config (from Notification access and notification listeners permission scope (/gms/policies/overview/amsterdam1#permission-groups)) or system UID is considered as a newly added MBA

MBAs with notification access or assigned as default notification listener MUST adhere to the following criteria:

- Only privileged or platform signed apps are approved as default Notification Listener Access packages.
- MUST follow Prominent disclosure and consent requirement (/gms/policies/domains/mba#disclosure-consent).
- MUST point the user to the Notification Listener Settings UI as a means of controlling access (**Settings** > ... > **Special app access** > **Notification access** or **Device & app notifications**). If this setting is denied, notifications MUST NOT be accessed. Exemptions to this requirement are granted for certain default handlers as described in the permitted use cases page.





- Any such disclosure or link to settings MUST show before the app accesses notification data.

The permission group, permitted uses, and exemptions are available in the Notification Access and Notification Listeners Policy Supplement (/gms/policies/overview/amsterdam1).

13.2.7 Accessibility and UI automation policy

For new launching DEVICEs submitted through Android Partner Approvals (APA) portal on or after:

- **June 10, 2021**: partners MUST fix accessibility and UI automation (/gms/policies/domains/introduction#definitions) policy violations to every MBA in the software build (/gms/policies/domains/introduction#definitions).

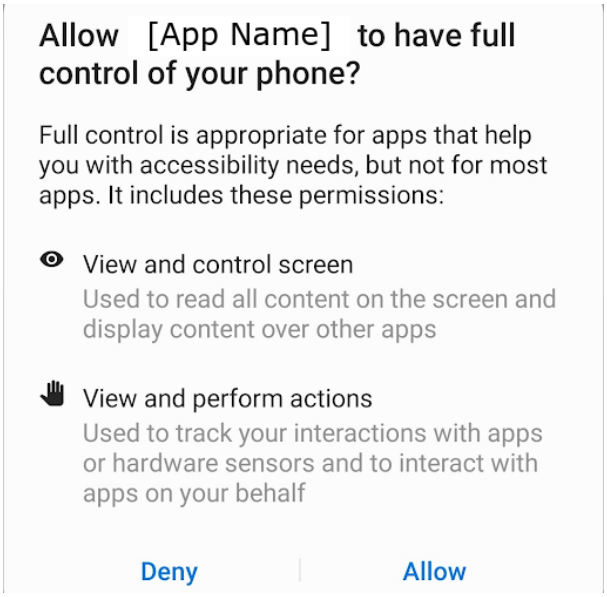
This includes:

- Every MBA (/gms/policies/domains/introduction#definitions) in the BUILD (if it's for a new launching DEVICE (/gms/policies/domains/introduction#definitions))

- Any newly added MBA in the build (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))
- Existing MBA (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) adding any new permission/config (from Accessibility and UI Automation permission scope (/gms/policies/overview/amsterdam2#permission-groups)) or system UID is considered as a newly added MBA

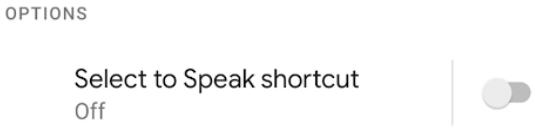
MBAs pre-granting accessibility or UI automation (/gms/policies/domains/introduction#definitions) capabilities to themselves or other apps MUST adhere to the following criteria:

- MUST follow Prominent disclosure and consent requirement (/gms/policies/domains/mba#disclosure-consent).
- MUST give the user control via a UI as below:
  - MBA pregranted accessibility access at build time MUST show below UI during device setup or when the MBA started the first time (before the MBA uses any accessibility services).
  - MBA pregranting accessibility access to itself or to another app at runtime MUST show below UI before the app is pregranted accessibility and before it uses any accessibility services.



  - Consent outcome MUST be reflected in Secure setting and System settings (Settings > Accessibility).
- MBA MUST be listed under System Accessibility Settings UI as a means of controlling access (Settings > Accessibility). If this setting is denied, accessibility privileges

MUST NOT be granted to the app.



- `RETRIEVE_WINDOW_CONTENT` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.RETRIEVE_WINDOW_CONTENT)
  MUST NOT be used on release key builds.

The permission group, permitted/prohibited use cases, and exemptions are available in the Accessibility and UI Automation Policy Supplement (/gms/policies/overview/amsterdam2).

13.2.8 App and network usage statistics policy

For new launching DEVICEs submitted through Android Partner Approvals (APA) portal on or after:

- **June 10, 2021**: partners MUST fix App and Network Usage Statistics Policy (/gms/policies/overview/amsterdam3) violations to every MBA in the software build (/gms/policies/domains/introduction#definitions).

This includes:

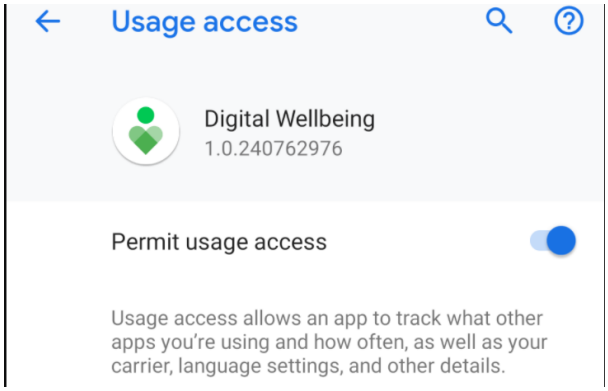
- Every MBA (/gms/policies/domains/introduction#definitions) in the BUILD (if it's for a new launching DEVICE (/gms/policies/domains/introduction#definitions))
- Any newly added MBA (/gms/policies/overview/gms-requirements#definitions) in the build (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))
- Existing MBA (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) adding any new permission/config (from the app and network usage statistics permission scope (/gms/policies/overview/amsterdam3#permission-groups)) or system UID is considered as a newly added MBA.

MBAs pregranted access to app or network usage statistics MUST adhere to the following criteria:

- MUST follow Prominent disclosure and consent requirement (/gms/policies/domains/mba#disclosure-consent) and prominently disclose any use of

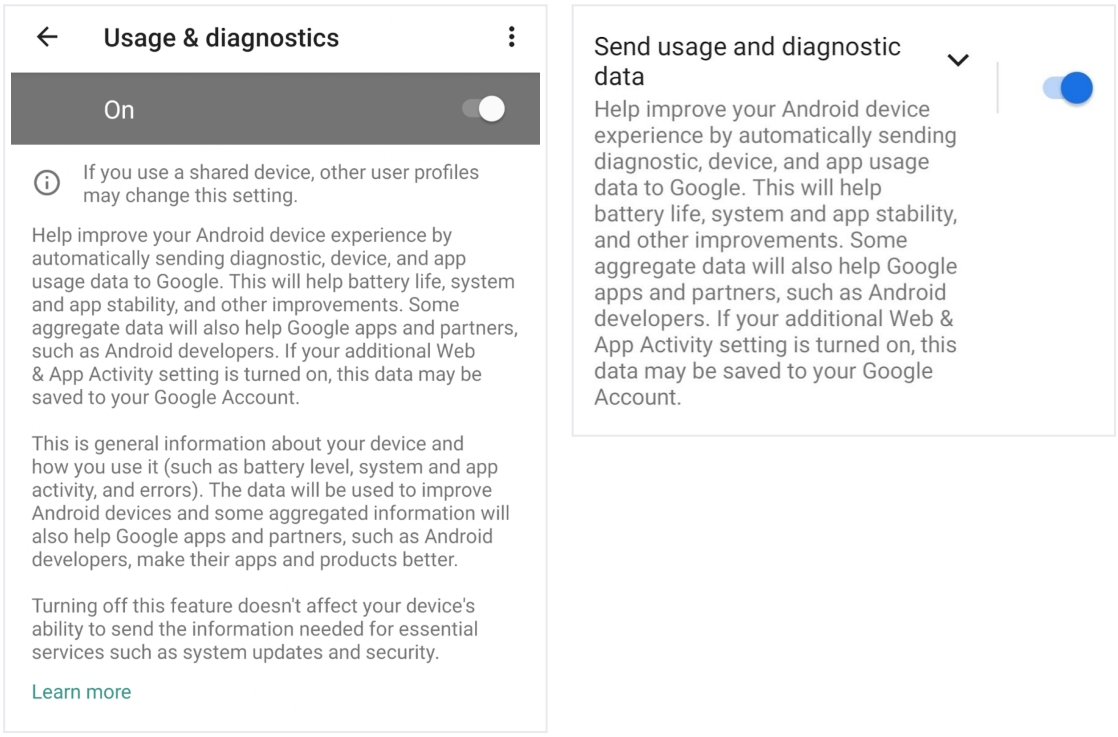
network or package usage information.

- MBAs with the `PACKAGE_USAGE_STATS` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.PACKAGE_USAGE_STATS) permission granted through `OPSTR_GET_USAGE_STATS` (https://developer.android.com/reference/android/app/AppOpsManager#OPSTR_GET_USAGE_STATS) app-op, MUST point the user to **Usage access** settings as a means of controlling access (**Settings** > **Apps** > **Advanced** > **Special app access** > **Usage access**). If this setting is denied by the user, the access MUST NOT be re-enabled automatically.



- For MBAs that obtain access without an explicit user grant via settings UI **Special app access** > **Usage access** (i.e. pre-grants of `PACKAGE_USAGE_STATS` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.PACKAGE_USAGE_STATS) or `OBSERVE_APP_USAGE` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.OBSERVE_APP_USAGE) or `GET_APP_OPS_STATS` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.GET_APP_OPS_STATS) or `READ_NETWORK_USAGE_HISTORY` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.READ_NETWORK_USAGE_HISTORY) or `DUMP` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.DUMP)) OEMs MUST ensure that there's a prominent disclosure and a mechanism to opt-out of collection before the app is used, as well as a mechanism to opt out at a later time. If the user opts out, usage data MUST NOT be accessed.
    - For example a pointer to settings app UI or dedicated settings in the app for allowing opt out, for example, following Google Play Services Usage & Diagnostics toggle (which controls the use of DUMP-gated data).

- Any such prominent disclosure and link to settings MUST show before the app accesses app and network usage statistics data.
- Exemptions to this requirement are granted for certain default handlers as described in the permitted use cases page (/gms/policies/overview/amsterdam3).



The permission group, permitted/prohibited use cases, and exemptions are available in the App and Network Usage Statistics Policy Supplement (/gms/policies/overview/amsterdam3).

13.2.9 Forced or keep enabled apps policy

For new launching DEVICEs submitted through Android Partner Approvals (APA) portal on or after:

- **Sep 8, 2021**: partners MUST fix forced or keep enabled apps policy violations to every MBA in the software build (/gms/policies/domains/introduction#definitions).

This includes:

- Every MBA (/gms/policies/domains/introduction#definitions) in the BUILD (if it's for a new launching DEVICE (/gms/policies/domains/introduction#definitions))
- Any new forced-enabled MBA (/gms/policies/domains/introduction#definitions) in the build (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))

Forced or keep enabled apps (/gms/policies/domains/introduction#definitions) MUST meet the following criteria:

- They MUST be preloaded full-version apps (not stubs (/gms/policies/domains/introduction#definitions)).
- MUST provide functionality described within the list of permitted use-cases (/gms/policies/overview/amsterdam4#permitted-use-cases)
- MUST NOT contain any functionality outside of the permitted use-cases (/gms/policies/overview/amsterdam4#permitted-use-cases), that cannot be separated into a different app running without the forced-enabled privilege.

The system **Settings** app MUST present an option to disable or uninstall every MBA that isn't explicitly allowed by this policy. And when an app is disabled by the user, that app MUST stay disabled until one of following things occurs:

- The app is manually reenabled by the user.
- The app is manually reinstalled by the user.
- The system is factory reset.

The permission group, permitted/prohibited use cases, and approval process are available in Force or Keep Enabled Apps Policy Supplement (/gms/policies/overview/amsterdam4).

13.2.10 Shared System UIDs policy

For New DEVICEs (/gms/policies/domains/introduction#definitions) launching with Android 12 or higher and submitted on the APA portal on or after:

- **Sep 8, 2021**: partners MUST fix Shared System UIDs policy violations on every MBA (/gms/policies/domains/introduction#definitions) in the software build (/gms/policies/domains/introduction#definitions).

Additionally, the following types of MBAs violating this policy MUST NOT be used on Android 9 (Pie) or higher PRODUCTs (/gms/policies/domains/introduction#definitions) on or after:

- **Sep 8, 2021**:
    - Any newly added MBA (/gms/policies/domains/introduction#definitions) in the build (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))

- Existing MBA (/gms/policies/domains/introduction#definitions) (if it's for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) that wasn't sharing system UIDs (/gms/policies/domains/introduction#definitions) earlier but started sharing any system UIDs (/gms/policies/domains/introduction#definitions) after the enforcement date is considered as a newly added MBA.

MBAs applying `sharedUserId` (https://developer.android.com/guide/topics/manifest/manifest-element#uid) with System UIDs (/gms/policies/domains/introduction#definitions) or custom System UIDs (/gms/policies/domains/introduction#definitions) MUST fulfill the following criteria:

- The primary functionality of the MBA MUST be required for the system to function. It's STRONGLY RECOMMENDED that the app doesn't contain any nonessential functionality that can't be separated into a different app running without the System UIDs (/gms/policies/domains/introduction#definitions).
- Preloaded full-version apps (not stubs (/gms/policies/domains/introduction#definitions)).
- MBAs may use System UIDs (/gms/policies/domains/introduction#definitions) only in such cases where there's no alternative way to obtain the same capability.
- If an MBA is accessing any Android API that's gated by a permission for which there's an existing MBA policy (/gms/policies/domains/mba), then it MUST follow the same policies including going through respective approval/exemption processes. [For example, when accessing APIs gated by dangerous permissions (https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions), apps must follow user data policy (/gms/policies/domains/mba#mba-user-data) and pregrant permissions policy (/gms/policies/domains/mba#mba-pregrant-permissions) and apply for the exception as per the pregrant permissions exemptions (/gms/policies/overview/pregrant-exemptions) process.]
    - Where required by the respective MBA policies, a toggle for requested permissions MUST be provided through an Alternative UI (/gms/policies/domains/mba#alt-UI-screen). Data obtained through the permission MUST NOT be accessed until the user has explicitly accepted through the Alternative UI (/gms/policies/domains/mba#alt-UI-screen). If the user opts out, usage data MUST NOT be accessed and the access MUST NOT be re-enabled automatically. Exemptions to this requirement are granted for certain use cases as described in the permitted use cases article (/gms/policies/overview/amsterdam5).
- If an MBA uses any AOSP API that's protected by any of the AOSP permissions protection levels (/gms/policies/domains/introduction#definitions) (e.g. dangerous, app-op,

privileged, signature, normal etc.), then the corresponding permissions MUST be declared regardless of whether any shared system UID is used to get that permission. This is required for permission tracking, auditing, policy validation, and enforcement purposes.

The prohibited use cases are available in Shared System UIDs Policy Supplement (/gms/policies/overview/amsterdam5).

13.2.11 Permission sharing policy

The following policy applies to permission sharing apps (/gms/policies/domains/introduction#definitions):

1. Any MBA (service app) exposing personal or sensitive capabilities (/gms/policies/domains/introduction#definitions) to another app (caller app) is STRONGLY RECOMMENDED to do one of the following:
    - Require the caller to hold all permissions required by the service to execute the service call.
    - Require the user's approval (for example, explicit permission or prominent notice) before accessing exposed capabilities (/gms/policies/domains/introduction#definitions), each time the service is called, unless the user explicitly requests a more permanent grant (for example, don't ask me again). Any such permanent grant should be revocable.
2. For any service interface that uses capabilities granted without explicit user permission (for example, pregranted runtime permissions), it is STRONGLY RECOMMENDED to do one of the following:
    - The caller meets the same conditions for exemption from obtaining user permission as the service.
    - The caller obtains user permission according to the requirements in the point 1 above.

    Exemptions are defined in the Pregrant permissions policy (/gms/policies/domains/mba#mba-pregrant-permissions) or the above conditions for pregranting of special access permissions.

13.3 MBA security policies

13.3.1 Minimum target SDK

- Every new Android platform release introduces changes that bring significant security improvements, but some of these changes apply only to apps that explicitly declare support through their `targetSdkVersion` manifest attribute. To take advantage of the platform's latest security features, all preloaded apps on DEVICEs running Android 9 and DEVICEs upgraded to Android 10 are STRONGLY RECOMMENDED to target SDK 26 or later.
- All apps preloaded on DEVICEs launching with Android 10 MUST target API level 28 or higher.
- All apps preloaded on DEVICEs launching with Android 11 or higher MUST target API level 29 or higher.

Android 12 introduces foreground service restrictions to improve performance and ensures a better experience for users.

For IR builds submitted through the Android Partner Approvals portal on or after:

- **December 15, 2022** : all MBAs MUST target Android 12 (targetSDK 31) or higher if:
  - using `FOREGROUND_SERVICE` permission **AND**
  - preloading on DEVICEs launching with Android 12 or higher.

For IR builds submitted through the Android Partner Approvals portal on or after:

- **December 15, 2022** : all MBAs MUST target Android 11 (targetSDK 30) or higher if:
  - NOT using `FOREGROUND_SERVICE` permission **AND**
  - preloading on DEVICEs launching with Android 12 or higher.

Start new requirements for 12

Exemption for vendor apps

For DEVICEs launching with Android 12 or higher, a preloaded vendor apps may be exempted from the policy if:

- It's distributed as a pre-compiled binary by the SoC vendor as a standalone APK or an APK embedded in APEX.
- It's a Headless app and does not have any user-interface components.

- It's preloaded in the `/vendor` partition and its function is only for supporting the capability of underlying SoC hardware.
- It uses the same package name across all OEMs.
- It's pre-signed by the SoC vendor or it's signed with the platform key by OEMs.
    - If it's signed with the platform key, it complies with the platform signing policy (/gms/policies/domains/platform-policies#platform-signing).
- It isn't updated by any app store or other app at runtime.

End new requirements

13.3.2 Dynamic code loading (DCL) policy

DEVICEs MUST allow only MBAs that dynamically load code from a trusted source (such as a manufacturer-owned server or a trusted partner's server like Google Play) over a secure connection (such as TLS), and only after the integrity of the code is verified.

13.3.3 App installation policy

DEVICEs (/gms/policies/domains/introduction#definitions) MUST allow MBAs (/gms/policies/domains/introduction#definitions) to install other apps only when those MBAs don't have the risk or history of becoming a notable source of downloading or installing other PHAs.

For DEVICEs (/gms/policies/domains/introduction#definitions) submitted through Android Partner Approvals (APA) portal on or after:

- **June 10, 2021**: partners MUST fix “App installation policy” violations to every MBA in the software build. This includes:
    - Every MBA (/gms/policies/domains/introduction#definitions) in the BUILD (if it's for a new launching DEVICE (/gms/policies/domains/introduction#definitions))
    - Any newly added MBA (/gms/policies/domains/introduction#definitions) in the build (if it is for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))
    - Existing MBA (if it is for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) adding any new permission (from App installation permission scope

(/gms/policies/overview/amsterdam6#permission-in-scope)) or System UID will be considered as a newly added MBA.

MBA installing other apps by any means MUST provide transparency and control to its users regarding the apps it installs on the device per the following requirements :

- App installers :
  - MUST provide app name, description, version number, download size, name, and contact information of the developer (the provider of the app) for every app they've installed. This should be provided either directly through a user interface or a link where users can see such information (for example, an app store).
    - All app installations MUST either show a notification or a persistent notice (/gms/policies/domains/introduction#definitions) where names of the installer and the installed app MUST be mentioned. Such notifications and notices MUST NOT be closed automatically.
    - During device setup or device provisioning (e.g. carrier installers installing apps during SIM insertion), app installers MAY alternatively show the number of apps that they've installed with a link to the detailed information (for example, in an app store), as specified above.
    - Headless (/gms/policies/domains/introduction#definitions) app installers MUST show the above information about the apps they installed via a persistent notice (/gms/policies/domains/introduction#definitions). This persistent notice MUST provide the installer name to establish provenance.
  - Are STRONGLY RECOMMENDED to show a progress bar (https://developer.android.com/training/notify-user/build-notification#progressbar) of the app being installed and a cancel button to cancel the app download while download is in progress.
  - Are STRONGLY RECOMMENDED to provide content ratings of installed apps, showing the minimum maturity level of content in apps.
  - MUST NOT declare `GRANT_RUNTIME_PERMISSIONS` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.GRANT_RUNTIME_PERMISSIONS) and `INSTALL_GRANT_RUNTIME_PERMISSIONS` (https://cs.android.com/android/platform/superproject/+/master:frameworks/base/core/res/AndroidManifest.xml?q=android.permission.INSTALL_GRANT_RUNTIME_PERMISSIONS) except when it has the ability to support Instant apps (https://developer.android.com/topic/google-play-instant) (those apps that run

without explicit installing). Only permissions with Protection level Instant (https://developer.android.com/reference/android/R.attr#protectionLevel) can be granted to Instant apps (https://source.android.com/compatibility/cts/cts-instant).

- MUST NOT install it again on the device automatically without explicit user control until the device is factory reset.
- MUST ensure that all installed apps are covered by a privacy and security policy. Privacy policies MUST explain how they treat user personal data (/gms/policies/domains/mba#personal-sensitive-info) and protect user's privacy when using that App Installer. These policies MUST be published with publicly accessible media such as a website.

- Backup, restore or device-to-device transfer systems :
  - MUST restore ONLY those apps that the user previously backed up or transferred from their devices
  - May do so without a dedicated UI for every app being restored.
  - MUST notify the user of the number of apps transferred.
- The System Settings app MUST present an option to uninstall every app that is installed or uninstall any app update if the app cannot be uninstalled.

Note: If an app installs any app without the user explicitly choosing them to be installed then such installed apps are considered to be MBAs (/gms/policies/domains/introduction#definitions) and hence would be subject to security and privacy policies defined under MBA policies (/gms/policies/domains/mba) and Platform policies (/gms/policies/domains/platform-policies).

The permission group, permitted/prohibited uses, and exemptions are available in the App Installation Policy Supplement (/gms/policies/overview/amsterdam6).

Important: Violation of this policy may be enforced via an entity ban. See MBA Entity Ban Enforcement (/gms/policies/overview/entity-ban) for more information.

13.3.4 Security vulnerabilities policy

Code MUST NOT introduce security vulnerabilities.

For every software build (/gms/policies/domains/introduction#definitions) for GMS devices submitted on the Android Partner Approvals (APA) portal on or after:

- **October 2020**: partners MUST fix all security vulnerabilities within 90 days of the date of the vulnerability disclosures. These fixes MUST be applied to every MBA in the software build.

For examples of security vulnerabilities, see the App Security Improvement Program (https://developer.android.com/google/play/asi.html#campaigns), Android Security Bulletin (https://source.android.com/security/bulletin), or published CVEs.

For more information about security vulnerabilites, see Device manufacturer best practices (https://support.google.com/androidpartners_security/answer/9054028) and Application developer best practices sites (https://support.google.com/androidpartners_security/answer/9055411).

For more information about this policy and enforcement, see Security Vulnerabilities Enforcement (/gms/policies/overview/security-vulnerabilities) for additional details.

Important: Violation of this policy may be enforced via an entity ban. See MBA Entity Ban Enforcement (/gms/policies/overview/entity-ban) for more information.

13.3.5 Valid V2+ signature for preloaded APKs

[GMS-13.3.5-001] For PRODUCTs that launch with or upgrade to Android 11 or higher, if a preloaded APK file targets API level 30 or higher, it MUST be signed and verifiable with the APK Signature scheme v2 or higher. Any native libraries embedded in it MUST be uncompressed and page aligned before signing.

See Valid V2+ signature guide (/gms/building/integrating/jni-libs) for more details about how to uncompress JNI libs and dex files and impact on the devices.

13.3.6 Debug certificate policy

For DEVICEs (/gms/policies/domains/introduction#definitions) submitted through Android Partner Approvals (APA) portal on or after:

- **June 10, 2021**: partners MUST fix Debug certificate policy violations to every MBA in the software build (/gms/policies/domains/introduction#definitions).

This includes:

- Every MBA (/gms/policies/domains/introduction#definitions) in the BUILD (if it's for a new DEVICE (/gms/policies/domains/introduction#definitions))
- Any newly added MBA (/gms/policies/domains/introduction#definitions) in the build (if it is for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions))
- Existing MBA (if it is for a previously approved PRODUCT (/gms/policies/domains/introduction#definitions)) adding any new debug certificate will be considered as a newly added MBA.

Android's debug certificate is created by the build tools and is insecure by design. Debug certificate is used to automatically sign apps in development environments, hence enabling other apps signed with that certificate to access its functionality without requesting any permission. As per Debug certificate policy:

- MBAs signed with any debug certificate MUST NOT be allowed on platform release key signed builds.
- To get MBAs signed, MBAs are STRONGLY RECOMMENDED to follow APK Signature Scheme v3 (https://source.android.com/security/apksigning/v3) on Android 9 and 10 devices and APK Signature Scheme v4 (https://source.android.com/security/apksigning/v4) and above on Android 11 and above devices.

Note: Further instructions are provided for developers at the Sign your app (https://developer.android.com/studio/publish/app-signing) page that shows how to properly sign an app for release.

Last updated 2021-10-04 UTC.