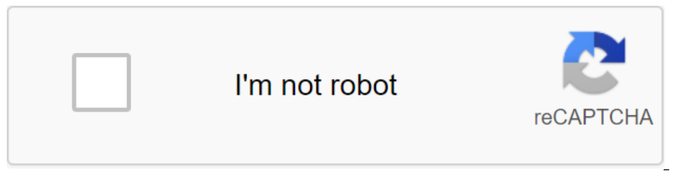


Continue

# Android gsi rom

A general system image (GSI) is a system image configured for Android devices. GSI can be considered a clean Android implementation with the unmodified Android Open Source Project (AOSP), which works smoothly on any Android device running Android 8.1 or above. GSI is used to run VTS and CTS-on-GSI tests. To make sure that the device running the latest version of Android implements the vendor interface correctly, you need to replace the system image of the Android device with GSI, and then test the device with a provider testing package (VTS) and a suite of compatibility tests (CTS). Note: The GSI content described in this article is available for Android OEM and ROM developers; Android app developers can visit developer.android.com for GSI details for app developers. Before you use GSI, read the following sections to learn more about GSI configurations (and permitted differences), types (Android GSI and outdated GSI), and vendor and VNDK dependencies. Then download and create GSI for your target device, and then clean the GSI for your Android device. GSI Configurations and Differences Current Android GSI has the following configuration: Treble - GSI fully supports the HIDL-scheme changes introduced in Android 8.0 (also known as Treble), including support for HIDL interfaces. You can use GSI on any Android device that uses the HIDL vendor interface (see Architecture Resources for more information). Download check at launch - GSI does not include a launch check solution such as vboot 1.0 or AVB. To write a GSI brush on a device with Android 9 or earlier, the device must have a way to disable the download time check. File system - GSI uses the ext4 file system. Section layout - GSI uses the system section layout as the root. The current Android GIS contains the following major differences: processor architecture - support for various processor instructions (ARM, x86, etc.) and CPU bits (32-bit or 64-bit). The GSI used in the GSI compliance test to test Treble compliance depends on the Android version on which the device was on at the time of its release. Type Build Target devices with Android 10 when releasing aosp_ $arch-user Devices with Android 9 when released aosp_$arch-userdebug with Android 8.0 or Android 8.1 when released aosp_ aosp_ All GSI arch_ab-userdebug is built on the basis of Android 10 code, and each processor scheme has an appropriate GSI binary (see list of building goals in GSI). Android 10 GSI changes devices from Android 10 at the time of release should use Android 10 GSI to test compliance. This GSI has the following major changes compared to the previous GSI: User Build - User for this GSI is based on Android 10. In Android 10, the custom version of GSI is available in CTS-on-GSI/VTS compliance testing. For more information, contact VTS testing with Debug Ramdisk. Non-rare format - GSI aosp_$arch is built in a non-rare format. If necessary, you can use img2simg to convert non-rare GSI into a rare format. system as root - - GSI aosp_ arch_a that the $100,000 construction goal had been terminated. For devices that upgrade to Android 10 with Android 8 or 8.1 and use ramdisk and non-system-as-root, use the old GSI aosp_ $arch.ab. The upgraded init in ramdisk supports the OEM system.img with the system as a root layout. To use Android GSI to create your targets with Android 9 or 10 devices when released through the CTS-on-GSI test. The old name GSI GSI has _ab (e.g. aosp_arm64_ab). These GSI are built from the android 10 source tree, but include the following backward compatible configurations for devices upgraded with Android 8 or 8.1: 32-bit user space plus a 32-bit binder interface - the 32-bit GSI can continue to use a 32-bit binder interface. 8.1 VNDK - The device can use the supplied 8.1 VNDK. Mount Catalog - Some outdated devices use catalog as pointers to mount (e.g.,-bluetooth, /firmware/radio, and /persist). To run a device with Android 8 or 8.1 on board through the CTS-on-GSI test release, build your targets using the old GSI. Note: If the device before Android 10 implements the Android 10 vendor interface and meets all the requirements presented in Android 10, use Android 10 GSI for VTS and CTS-on-GSI, not older than GSI. Android 9 GSI Changes Android 9 GSI The following major changes have occurred compared to previous GSI: The merger of GSI and Simulator - GSI is built on system images of emulatory products such as aosp_arm64 and aosp_x86. System-as-Root - In previous versions of Android, devices that don't support A/B upgrades can mount system images under/system catalog. In Android 9, the root of the system image is set as the root of the device. 64-bit binder interface - In Android 8.x, the 32-bit GSI uses a 32-bit binder interface. Android 9 doesn't support a 32-bit binder interface, so both the 32-bit GSI and the 64-bit GSI use a 64-bit binder interface. Providing VNDK - In Android 8.1, VNDK is optional. Starting with Android 9, VNDK is a must, so you have to set BOARD_VNDK_VERSION. Compatible Properties System - Android 9 supports access checks for compatible system properties: PRODUCT_COMPATIBLE_PROPERTY_OVERRIDE: . Android 9 Keymaster Change In earlier versions of Android devices sold by Keymaster 3 or earlier, should check whether the version information provided by the running system (ro.build.version.release and ro.build.version.security_patch) corresponds to the information about the version provided by the downloader. This information is usually derived from the download image header file. In Android 9 and later, this requirement was changed in order for the vendor to start GSI. In the Keymaster should not perform the review because the version information provided by GSI may not match the version information reported by the vendor downloader. For devices that are implementing Keymaster 3 or earlier, the vendor must change the implementation of Keymaster to skip the check (or upgrade to Keymaster 4). To learn more about Keymaster, see Supplier Binary and VNDK. VNDK. Devices upgraded to Android 10 have different upgrade paths, depending on the binary vendor version used on the device and the configuration associated with the VNDK used to create the device. The following table summarizes inherited GSI support for updated devices: Use a binary version of the case provider BOARD_VNDK_VERSION legacy of the GSI system Binary version legacy GSI support 0 8.0 (any) 10 No 1 8.1 (empty) 10 No 2 8.1 current 10 is the 2nd most common supported use case, where the old GSI supports devices built with Android 8.1 and with BOARD_VNDK_VERSION installed in current. Case 1 is not supported. In this case, the old GSI does not support devices with Android 8.1, but were built with BOARD_VNDK_VERSION built. These devices cannot be supported because their binary supplier relies on Android 8.1 non-VNDK common libraries that were not included in the old GSI. To make these devices compatible with an outdated GSI, you must include BOARD_VNDK_VERSION, but don't install BOARD_VNDK_RUNTIME_DISABLE (use case 2) or binary/vendor update to rely on the overall library in Android 10 (use case 3). Note: When you BOARD_VNDK_VERSION just for testing purposes, you can turn on BOARD_VNDK_RUNTIME_DISABLE. You shouldn't turn it on when you're creating a production image. Download GSI You can download the pre-built GSI from the AOSP Ci.android.com (CI) website. If GSI for your hardware platform is unavailable for download, see the following sections to learn more about how to build ASI for a specific purpose. Construction of GSI begins with Android 9, and each version of Android has a GSI branch called DESSERT-gsi on AOSP (for example, android10-gsi is a branch of GSI on Android 10). The GSI branch contains Android content that has all security patches and GSI patches are applied. To build ASI, download it from the GSI branch, and then select the purpose of the GSI build to customize the Android source tree. You can determine which version of GSI is appropriate for your device based on the next target assembly table. Once the build is complete, GSI becomes a system image (system.img) and appears under the output folder out/target/product/generic_arm64 folder. The build also displays vbmeta.img, which can be used to disable download time checks on devices that are tested when android is launched. For example, to build a GSI target aosp_arm64-userdebug on the GSI android10-gsi branch, run the next command. $repo init -u -b android10-gsi $REPO synchronization -cq $ source build/envsetup.sh $lunch aosp_arm64-userdebug $do-j4 Android GSI build targets Following GSI build targets for devices with Android 9 or later when released. Android 10 contains only four GSI products due to reduced architectural differences. GSI Processor Name Architecture Binder Interface Bit System-as-Root Build Target aosp_arm ARM 64 is a aosp_arm user aosp_arm-userdebug aosp_arm64 ARM64 64 aosp_arm64 user. User. aosp_x86 x86 64 is a aosp_x86-user aosp_x86-userdebug aosp_x86_64 x86-64 64 is aosp_x86_64-user aosp_x86_64-userdebug legacy GSI build a goal following outdated GSI build targets suitable for device upgrades with Android 8.0 or 8.1 to Android 10. Old GSI names contain _ab suffixes to distinguish them from Android 10 GSI names. GSI Processor Name Architecture Binder Interface Bit System-as-Root Build Target aosp_arm_ab ARM 32 is the aosp_arm_ab-user aosp_arm_64b_ab ARM 64 is aosp_arm_64b_ab-user aosp_arm64_ab ARM64 aosp_arm64_ab-userdebug. aosp_x86_ab x86 32 aosp_x86_ab-userdebug aosp_x86_64_ab x86-64 64 is a aosp_x86_64_ab-userdebug Note: These build goals may be removed in future Android releases. GSI cleaning requires Android devices to have different designs, so GSI cannot be like all devices with universal commands or universal instruction groups. You can ask the manufacturer of your Android device for detailed cleaning instructions, or you can refer to the following common steps: Make sure your device has the following conditions: Treble's support method for unlocking the device (so you can clean it with fastboot) to disable the check method when starting (e.g. vboot 1.0 or AVB) The device is unlocked, ensuring that you can swipe the device through fastboot (to make sure that you can swipe the device through fastboot) that you have the latest version of fastboot, build it based on the Android source tree. Deactivate the test at the start. Free the current section of the system and then brush the GSI to the system section. Removing user data and clear data from other necessary sections, such as user data sections and system sections. Reboot the device. For example, to clean the GSI to any Pixel device, start fastboot mode and unlock the loader. Fastbootd-enabled devices also need to start fastbootd, following the following command: $ fastboot reboot fastboot by cleaning vbmeta.img deactivated launch check (AVB): $fastboot-disable-flash-check vbmeta vbmeta.img empty section of the system, and then brush GSI into the system. Section: $fastboot erases the $20 fastboot flash system.img erases user data and erases data from other required sections (e.g., user data sections and sections of the system): $fastboot-w reboot: $fastboot reboot GSI brush on Android 10 devices with smaller sections of the system. The following error message may appear: Size 'system_a' FAILED (remote: 'Not enough room for section size') fastboot: Error: Team Failed You Can Use the Next Command to Remove the Product Section and make room for the system section. This provides additional room for GSI cleaning: $fastboot the logical section product_a the suffix _a must match the system section ID slot, such as the system_a in this example. To improve the positive contribution of GSI, Android welcomes you to contribute to the development of GSI. You can participate in the development and help improve GSI by creating GSI patches. DESSERT-gsi is not a development industry and is accepted only from AOSP. AOSP. The patches selected by the branch, so you need to send the GSI patches: send the patches to the main AOSP branch. Choose patches to present DESSERT-gsi at a merit-based level. Send the bug to have the merit-based patch reviewed. Report GSI errors or make other recommendations. Review the report error instructions and then review or submit the GSI error. Tip When you start with GSI, the bar navigation mode replaces the configuration through the vendor. You can change the navigation bar mode by putting forward the following adb command while running: adb exec-out cmd overlays allow-exclusive com.android.internal.systemui.navbar.mode, where the mode can be three buttons, two buttons, gestural, and so on. And so on. android 10 gsi rom. android 11 gsi rom. android 10 gsi rom download. android 8.1 gsi rom. android 9 gsi roms. android 10 gsi rom xda. android 9.0 gsi rom. rom gsi android 10 j7 prime