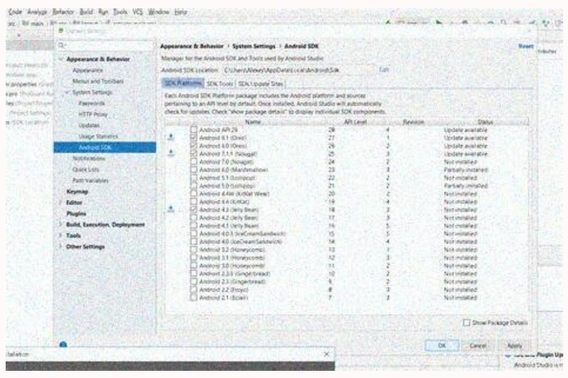


I am not robot!

What is 6.0 android version. Android version name with api level. Android 6.0.1 api version. Android version vs api level.

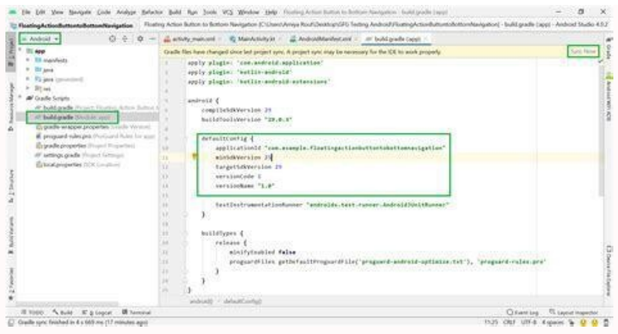
Stay organized with collections Save and categorize content based on your preferences. Android 6.0 gives you an opportunity to ensure your apps work with the next version of the platform. This release includes a number of APIs and behavior changes that can impact your app, as described in the API Overview and Behavior Changes. In testing your app with this release, there are some specific system changes that you should focus on to ensure that users have a good experience. This guide describes the what and how to test Android 6.0 features with your app. You should prioritize testing of these specific features, due to their high potential impact on your app's behavior: Permissions Doze and App Standby Auto Backup and Device Identifiers Testing Permissions The new Permissions model changes the way that permissions are allocated to your app by the user. Instead of granting all permissions during the install procedure, your app must ask the user for individual permissions at runtime. For users this behavior provides more granular control over each app’s activities, as well as better context for understanding why the app is requesting a specific permission. Users can grant or revoke the permissions granted to an app individually at any time. This feature of the release is most likely to have an impact on your app's behavior and may prevent some of your app features from working, or they may work in a degraded state. This change affects all apps running on the new platform, even those not targeting the new platform version.
The platform provides a limited compatibility behavior for legacy apps, but you should begin planning your app’s migration to the new permissions model now, with a goal of publishing an updated version of your app at the official platform launch. Test tips Use the following test tips to help you plan and execute testing of your app with the new permissions behavior. Identify your app’s current permissions and the related code paths. Test user flows across permission-protected services and data. Test with various combinations of granted/revoked permission. Use the adb tool to manage permissions from the command line: List permissions and status by group: adb shell pm list permissions -d -g Grant or revoke one or more permissions using the following syntax: adb shell pm [grant|revoke] ... Analyze your app for services that use permissions.
Test strategy The permissions change affects the structure and design of your app, as well as the user experience and flows you provide to users. You should assess your app’s current permissions use and start planning for the new flows you want to offer.
The official release of the platform provides compatibility behavior, but you should plan on updating your app and not rely on these behaviors. Identify the permissions that your app actually needs and uses, and then find the various code paths that use the permission-protected services. You can do this through a combination of testing on the new platform and code analysis. In testing, you should focus on opting in to runtime permissions by changing the app’s targetSdkVersion to API level 23. Test with various combinations of permissions revoked and added, to highlight the user flows that depend on permissions. Where a dependency is not obvious or logical you should consider refactoring or compartmentalizing that flow to eliminate the dependency or make it clear why the permission is needed. For more information on the behavior of runtime permissions, testing, and best practices, see Working with System Permissions developer. Testing Doze and App Standby The power saving features of Doze and App Standby limit the amount of background processing that your app can perform when a device is in an idle state or while your app is not in focus. The restrictions the system may impose on apps include limited or no network access, suspended background tasks, suspended Notifications, ignored wake requests, and alarms. To ensure that your app behaves properly with these power saving optimizations, you should test your app by simulating these low power states.



This guide describes the what and how to test Android 6.0 features with your app. You should prioritize testing of these specific features, due to their high potential impact on your app's behavior: Permissions Doze and App Standby Auto Backup and Device Identifiers Testing Permissions The new Permissions model changes the way that permissions are allocated to your app by the user. Instead of granting all permissions during the install procedure, your app must ask the user for individual permissions at runtime. For users this behavior provides more granular control over each app’s activities, as well as better context for understanding why the app is requesting a specific permission. Users can grant or revoke the permissions granted to an app individually at any time. This feature of the release is most likely to have an impact on your app's behavior and may prevent some of your app features from working, or they may work in a degraded state. This change affects all apps running on the new platform, even those not targeting the new platform version. The platform provides a limited compatibility behavior for legacy apps, but you should begin planning your app’s migration to the new permissions model now, with a goal of publishing an updated version of your app at the official platform launch. Test tips Use the following test tips to help you plan and execute testing of your app with the new permissions behavior. Identify your app’s current permissions and the related code paths. Test user flows across permission-protected services and data. Test with various combinations of granted/revoked permission. Use the adb tool to manage permissions from the command line: List permissions and status by group: adb shell pm list permissions -d -g Grant or revoke one or more permissions using the following syntax: adb shell pm [grant|revoke] ...
Analyze your app for services that use permissions. Test strategy The permissions change affects the structure and design of your app, as well as the user experience and flows you provide to users. You should assess your app’s current permissions use and start planning for the new flows you want to offer. The official release of the platform provides compatibility behavior, but you should plan on updating your app and not rely on these behaviors. Identify the permissions that your app actually needs and uses, and then find the various code paths that use the permission-protected services. You can do this through a combination of testing on the new platform and code analysis. In testing, you should focus on opting in to runtime permissions by changing the app’s targetSdkVersion to API level 23. Test with various combinations of permissions revoked and added, to highlight the user flows that depend on permissions. Where a dependency is not obvious or logical you should consider refactoring or compartmentalizing that flow to eliminate the dependency or make it clear why the permission is needed. For more information on the behavior of runtime permissions, testing, and best practices, see Working with System Permissions developer.
Testing Doze and App Standby The power saving features of Doze and App Standby limit the amount of background processing that your app can perform when a device is in an idle state or while your app is not in focus. The restrictions the system may impose on apps include limited or no network access, suspended background tasks, suspended Notifications, ignored wake requests, and alarms. To ensure that your app behaves properly with these power saving optimizations, you should test your app by simulating these low power states. Testing your app with Doze To test Doze with your app: Configure a hardware device or virtual device with an Android 7.0 (API level 24) system image.

Code name	Version number	Initial release date	API level	Security patches[1]
(No codename)[2]	1.0	September 23, 2008	1	Unsupported
(Internally known as "Petit Four")[2]	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut[3]	1.6	September 15, 2009	4	Unsupported
Eclair[4]	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo[5]	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread[6]	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb[7]	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich[8]	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean[9]	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat[10]	4.4 – 4.4.4	October 31, 2013	19 – 20	Supported[11] See clarification
Lollipop[12]	5.0 – 5.1.1	November 12, 2014	21 – 22	Supported
Marshmallow[13]	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat[14]	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
Oreo	**8.0**	**August 21, 2017**	**26**	**Supported**

This release includes a number of APIs and behavior changes that can impact your app, as described in the API Overview and Behavior Changes. In testing your app with this release, there are some specific system changes that you should focus on to ensure that users have a good experience. This guide describes the what and how to test Android 6.0 features with your app. You should prioritize testing of these specific features, due to their high potential impact on your app's behavior: Permissions Doze and App Standby Auto Backup and Device Identifiers Testing Permissions The new Permissions model changes the way that permissions are allocated to your app by the user. Instead of granting all permissions during the install procedure, your app must ask the user for individual permissions at runtime. For users this behavior provides more granular control over each app’s activities, as well as better context for understanding why the app is requesting a specific permission. Users can grant or revoke the permissions granted to an app individually at any time. This feature of the release is most likely to have an impact on your app's behavior and may prevent some of your app features from working, or they may work in a degraded state. This change affects all apps running on the new platform, even those not targeting the new platform version.

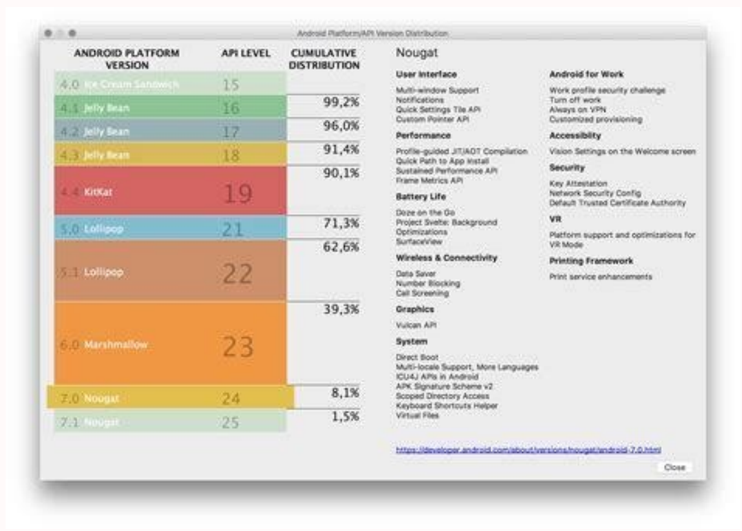


Android version vs api level.

Stay organized with collections Save and categorize content based on your preferences. Android 6.0 gives you an opportunity to ensure your apps work with the next version of the platform. This release includes a number of APIs and behavior changes that can impact your app, as described in the API Overview and Behavior Changes. In testing your app with this release, there are some specific system changes that you should focus on to ensure that users have a good experience. This guide describes the what and how to test Android 6.0 features with your app. You should prioritize testing of these specific features, due to their high potential impact on your app's behavior: Permissions Doze and App Standby Auto Backup and Device Identifiers Testing Permissions The new Permissions model changes the way that permissions are allocated to your app by the user.



This release includes a number of APIs and behavior changes that can impact your app, as described in the API Overview and Behavior Changes. In testing your app with this release, there are some specific system changes that you should focus on to ensure that users have a good experience. This guide describes the what and how to test Android 6.0 features with your app. You should prioritize testing of these specific features, due to their high potential impact on your app's behavior: Permissions Doze and App Standby Auto Backup and Device Identifiers Testing Permissions The new Permissions model changes the way that permissions are allocated to your app by the user. Instead of granting all permissions during the install procedure, your app must ask the user for individual permissions at runtime. For users this behavior provides more granular control over each app's activities, as well as better context for understanding why the app is requesting a specific permission. Users can grant or revoke the permissions granted to an app individually at any time. This feature of the release is most likely to have an impact on your app's behavior and may prevent some of your app features from working, or they may work in a degraded state. This change affects all apps running on the new platform, even those not targeting the new platform version. The platform provides a limited compatibility behavior for legacy apps, but you should begin planning your app's migration to the new permissions model now, with a goal of publishing an updated version of your app at the official platform launch. Test tips Use the following test tips to help you plan and execute testing of your app with the new permissions behavior. Identify your app's current permissions and the related code paths. Test user flows across permission-protected services and data. Test with various combinations of granted/revoked permission. Use the adb tool to manage permissions from the command line: List permissions and status by group: adb shell pm list permissions -d -g Grant or revoke one or more permissions using the following syntax: adb shell pm [grant|revoke] ... Analyze your app for services that use permissions. Test strategy The permissions change affects the structure and design of your app, as well as the user experience and flows you provide to users. You should assess your app's current permissions use and start planning for the new flows you want to offer. The official release of the platform provides compatibility behavior, but you should plan on updating your app and not rely on these behaviors. Identify the permissions that your app actually needs and uses, and then find the various code paths that use the permission-protected services. You can do this through a combination of testing on the new platform and code analysis. In testing, you should focus on opting in to runtime permissions by changing the app's targetSdkVersion to API level 23. Test with various combinations of permissions revoked and added, to highlight the user flows that depend on permissions. Where a dependency is not obvious or logical you should consider refactoring or compartmentalizing that flow to eliminate the dependency or make it clear why the permission is needed. For more information on the behavior of runtime permissions, testing, and best practices, see Working with System Permissions developer. Testing Doze and App Standby The power saving features of Doze and App Standby limit the amount of background processing that your app can perform when a device is in an idle state or while your app is not in focus.
The restrictions the system may impose on apps include limited or no network access, suspended background tasks, suspended Notifications, ignored wake requests, and alarms.



Android version vs api level.

Stay organized with collections Save and categorize content based on your preferences. Android 6.0 gives you an opportunity to ensure your apps work with the next version of the platform. This release includes a number of APIs and behavior changes that can impact your app, as described in the API Overview and Behavior Changes. In testing your app with this release, there are some specific system changes that you should focus on to ensure that users have a good experience. This guide describes the what and how to test Android 6.0 features with your app. You should prioritize testing of these specific features, due to their high potential impact on your app's behavior: Permissions Doze and App Standby Auto Backup and Device Identifiers Testing Permissions The new Permissions model changes the way that permissions are allocated to your app by the user. Instead of granting all permissions during the install procedure, your app must ask the user for individual permissions at runtime. For users this behavior provides more granular control over each app's activities, as well as better context for understanding why the app is requesting a specific permission. Users can grant or revoke the permissions granted to an app individually at any time. This feature of the release is most likely to have an impact on your app's behavior and may prevent some of your app features from working, or they may work in a degraded state. This change affects all apps running on the new platform, even those not targeting the new platform version. The platform provides a limited compatibility behavior for legacy apps, but you should begin planning your app's migration to the new permissions model now, with a goal of publishing an updated version of your app at the official platform launch. Test tips Use the following test tips to help you plan and execute testing of your app with the new permissions behavior. Identify your app's current permissions and the related code paths. Test user flows across permission-protected services and data. Test with various combinations of granted/revoked permission. Use the adb tool to manage permissions from the command line: List permissions and status by group: adb shell pm list permissions -d -g Grant or revoke one or more permissions using the following syntax: adb shell pm [grant|revoke] ... Analyze your app for services that use permissions. Test strategy The permissions change affects the structure and design of your app, as well as the user experience and flows you provide to users. You should assess your app's current permissions use and start planning for the new flows you want to offer. The official release of the platform provides compatibility behavior, but you should plan on updating your app and not rely on these behaviors. Identify the permissions that your app actually needs and uses, and then find the various code paths that use the permission-protected services. You can do this through a combination of testing on the new platform and code analysis. In testing, you should focus on opting in to runtime permissions by changing the app's targetSdkVersion to API level 23. Test with various combinations of permissions revoked and added, to highlight the user flows that depend on permissions. Where a dependency is not obvious or logical you should consider refactoring or compartmentalizing that flow to eliminate the dependency or make it clear why the permission is needed.
For more information on the behavior of runtime permissions, testing, and best practices, see Working with System Permissions developer. Testing Doze and App Standby The power saving features of Doze and App Standby limit the amount of background processing that your app can perform when a device is in an idle state or while your app is not in focus. The restrictions the system may impose on apps include limited or no network access, suspended background tasks, suspended Notifications, ignored wake requests, and alarms. To ensure that your app behaves properly with these power saving optimizations, you should test your app by simulating these low power states. Testing your app with Doze To test Doze with your app: Configure a hardware device or virtual device with an Android 7.0 (API level 24) system image. Connect the device to your development machine and install your app. Run your app and leave it active. Simulate the device going into Doze mode by running the following commands: $ adb shell dumpsys battery unplug $ adb shell dumpsys deviceidle step $ adb shell dumpsys deviceidle -h Observe the behavior of your app when the device is re-activated. Make sure it recovers gracefully when the device exits Doze. Testing apps with App Standby To test the App Standby mode with your app: Configure a hardware device or virtual device with an Android 7.0 (API level 24) system image. Connect the device to your development machine and install your app.
Run your app and leave it active. Simulate the app going into standby mode by running the following commands: $ adb shell am broadcast -a android.os.action.DISCHARGING $ adb shell am set-idle true Simulate waking your app using the following command: $ adb shell am set-idle false Observe the behavior of your app when it is woken. Make sure it recovers gracefully from standby mode.
In particular, you should check if your app's Notifications and background jobs continue to function as expected. Auto Backup for Apps and Device-Specific Identifiers If your app is persisting any device-specific identifiers, such as Google Cloud Messaging registration ID, in internal storage, make sure to follow best practices to exclude the storage location from auto-backup, as described in Back Up User Data with Auto Backup. Stay organized with collections Save and categorize content based on your preferences.
This page provides release information about the SDK packages available for download from the SDK Manager, in the SDK Platforms tab. Each SDK Platform version includes the following packages: The Android SDK Platform package. This is required to compile your app for that version.
Several System Image packages. At least one of these is required to run that version on the Android Emulator. Each platform version includes a system image for each supported form factor (handsets, Android TV, and Android Wear). Each form factor might offer variations to match your computer's processor architecture (such as Intel x86 and ARM EABI). System images labeled Google APIs include access to Google Play services and those labeled Google Play also include the Google Play Store. The Sources for Android package. This includes the source files for the platform. Android Studio may show lines of code from these files while you debug your app. The revision numbers listed below are for the Android SDK Platform package only.
The system images may receive separate updates, usually to resolve bugs with the emulator. There are no release notes for the system images, but you should always keep them up to date. Important: To see the most recent Android system components in the Android SDK Manager, you must first install the most recent Android SDK Command-Line tools package. Android 14 (API level 34) For details about the platform changes, see the Android 14 documentation. Android 13 (API level 33) For details about the platform changes, see the Android 13 documentation.
Android 12 (API levels 31, 32) 12L feature drop (API level 32) For details about the platform changes, see the 12L documentation. Released to the stable channel (no longer in preview). Android 12 (API level 31) For details about the platform changes, see the Android 12 documentation. Released to the stable channel (no longer in preview) when Android 12 reached the Platform Stability milestone. Android 12 ATD system images This Automated Test Device (ATD) image is an Android system image that is optimized for headless automated tests. Early data indicates that tests that use this image should experience reduction in emulator process CPU and memory usage, and reduction in test wall time. It achieves these performance gains through: Removing most user-facing applications (for example Dialer, Settings, and SystemUI). Disabling hardware renderer drawing.
The image comes with two versions: Google APIs ATD which provides Google APIs, and AOSP ATD which provides pure AOSP experience. To learn more about running tests using ATDs, see Run tests using Automated Test Devices. Android 11 (API level 30) For details about the platform changes, see the Android 11 documentation. Android 10 (API level 29) For details about the platform changes, see Android 10 for Developers. This revision adds Android Automotive OS stubs. Android 9 (API level 28) For details about the platform changes, see Android 9 for developers. Released to the stable channel (no longer in preview). Android 8.1 (API level 27) For details about the platform changes, see Android 8.1 for developers. Released to the stable channel (no longer in preview). Android 8.0 (API level 26) For details about the platform changes, see Android 8.0 for developers. Released to the stable channel (no longer in preview). Android 7.1 (API level 25) For details about the platform changes, see Android 7.1 for developers. Incremental update. Released as the final Android 7.1.1 (no longer in preview). Dependencies: Android SDK Platform-Tools 25.0.1 or higher is required.
Android SDK Build-Tools 25.0.1 or higher is required. Incremental update. Released as Android 7.1.1 Developer Preview 2. For more information, see the Android 7.1 API Overview. Dependencies: Android SDK Platform-Tools 25.0.1 or higher is required.
Android SDK Build-Tools 25.0.1 or higher is required. Initial release for Android 7.1 (API level 25).
Released as Android 7.1 Developer Preview 1. For more information, see the Android 7.1 API Overview. Dependencies: Android SDK Platform-Tools 25.0.0 or higher is required. Android SDK Build-Tools 25.0.0 or higher is required. Android 7.0 (API level 24) For details about the platform changes, see Android 7.0 for developers. Initial release for Android 7.0 (API level 24). For more information, see the Android 7.0 API Overview. Dependencies: Android SDK Platform-tools r24 or higher is required. Android SDK Tools 24.0.0 or higher is required. Android 6.0 (API level 23) For details about the platform changes, see the Android 6.0 changes and Android 6.0 APIs. Fixed bugs in the layout rendering library used by Android Studio. Dependencies: Android SDK Platform-tools r23 or higher is required. Android SDK Tools 24.3.4 or higher is required. Initial release for Android 6.0 (API level 23). For more information, see the Android 6.0 API Overview. Dependencies: Android SDK Platform-tools r23 or higher is required. Android SDK Tools 24.3.4 or higher is required.
Android 5.1 (API level 22) For details about the platform changes, see the Lollipop overview and Android 5.1 API changes. Initial release for Android 5.1 (API level 22). For more information, see the Android 5.1 API Overview. Dependencies: Android SDK Platform-tools r22 or higher is required. Android SDK Tools 23.0.5 or higher is required. Android 5.0 (API level 21) For details about the platform changes, see the Lollipop overview and Android 5.0 API changes. Updated layouts in the Support Library and fixed various issues. Dependencies: Android SDK Platform-tools r21 or higher is required. Android SDK Tools 23.0.5 or higher is required. Initial release for Android 5.0 (API level 21). For more information, see the Android 5.0 API Overview.
Dependencies: Android SDK Platform-tools r21 or higher is required. Android SDK Tools 23.0.5 or higher is required. Android 4.4W (API level 20) This version makes KitKat available for Android Wear. Updated the rendering library. Dependencies: Android SDK Platform-tools r20 or higher is required. Android SDK Tools 23.0 or higher is required. Initial release for Android Wear. Dependencies: Android SDK Platform-tools r20 or higher is required. Android SDK Tools 23.0 or higher is required. Android 4.4 (API level 19) For details about the platform changes, see the KitKat overview and Android 4.4 API changes.
Maintenance release. The system version is 4.4.2. For more information, see the Android 4.4 API Overview. Dependencies: Android SDK Platform-tools r19 or higher is required. Android SDK Tools 22.3 or higher is recommended. Initial release. The system version is 4.4. For more information, see the Android 4.4 API Overview. Dependencies: Android SDK Platform-tools r19 or higher is required.
Android SDK Tools 22.3 or higher is recommended. Android 4.3 (API level 18) For details about the platform changes, see the Jelly Bean overview and Android 4.3 API changes. Maintenance update. The system version is 4.3. Dependencies: Android SDK Platform-tools r18 or higher is required. Android SDK Tools 22.0.4 or higher is recommended. Initial release. The system version is 4.3. Dependencies: Android SDK Platform-tools r18 or higher is required. Android SDK Tools 22.0.4 or higher is recommended.
Android 4.2 (API level 17) For details about the platform changes, see the Jelly Bean overview and Android 4.2 API changes. Maintenance update. The system version is 4.2.2. Dependencies: SDK Tools r21 or higher is required. Initial release. The system version is 4.2. Dependencies: SDK Tools r20 or higher is required. Android 4.1 (API level 16) For details about the platform changes, see the Jelly Bean overview and Android 4.1 API changes. Maintenance update. The system version is 4.1.2. Dependencies: SDK Tools r20 or higher is required. Maintenance update. The system version is 4.1.1. Dependencies: SDK Tools r20 or higher is required. Initial release. The system version is 4.1.0. Dependencies: SDK Tools r20 or higher is required. Android 4.0.3 (API level 15) Maintenance update. The system version is 4.0.4. Note: This system image includes support for emulator hardware graphics acceleration when used with SDK Tools r17 or higher. (more info) Dependencies: SDK Tools r17 or higher is required. Maintenance update. The system version is 4.0.3. Dependencies: SDK Tools r14 or higher is required. Initial release. The system version is 4.0.3. Dependencies: SDK Tools r14 or higher is required. Android 4.0 (API level 14) Maintenance update. The system version is 4.0.2. Dependencies: SDK Tools r14 or higher is required. Initial release. The system version is 4.0.1. Dependencies: SDK Tools r14 or higher is required. Android 3.2 (API level 13) Initial release. SDK Tools r12 or higher is recommended.
Android 3.1 (API level 12) Dependencies: Requires SDK Tools r12 or higher. Notes: Improvements to the platform's rendering library to support the visual layout editor in the ADT Eclipse plugin. This revision allows for more drawing features in ADT and fixes several bugs in the previous rendering library. It also unlocks several editor features that were added in ADT 12. Dependencies: Requires SDK Tools r11 or higher. Notes: Fixes an issue with the visual layout editor rendering library that prevented Android 3.1 from running in ADT. Dependencies: Requires SDK Tools r11 or higher. Android 3.0 (API level 11) Dependencies: Requires SDK Tools r12 or higher. Notes: Improvements to the platform's rendering library to support the visual layout editor in the ADT Eclipse plugin. This revision allows for more drawing features in ADT and fixes several bugs in the previous rendering library. It also unlocks several editor features that were added in ADT 12. Dependencies: Requires SDK Tools r10 or higher. Android 2.3.3 (API level 10) Dependencies: Requires SDK Tools r12 or higher. Notes: Improvements to the platform's rendering library to support the visual layout editor in the ADT Eclipse plugin. This revision allows for more drawing features in ADT and fixes several bugs in the previous rendering library. It also unlocks several editor features that were added in ADT 12. Dependencies: Requires SDK Tools r9 or higher. Android 2.3 (API level 9) Dependencies: Requires SDK Tools r8 or higher. (adsbygoogle = window.adsbygoogle || []).push({}); Android Version, Name and API level Android Name Version numbers API level Release date No name 1.0 1 September 23, 2008 No name 1.1 2 February 9, 2009 Cupcake 1.5 3 April 27, 2009 Donut 1.6 4 September 15, 2009 Eclair 2.0 – 2.1 5 – 7 October 26, 2009 Froyo 2.2 – 2.2.3 8 May 20, 2010 Gingerbread 2.3 – 2.3.7 9 – 10 December 6, 2010 Honeycomb 3.0 – 3.2.6 11 – 13 February 22, 2011 Ice Cream Sandwich 4.0 – 4.0.4 14 – 15 October 18, 2011 Jelly Bean 4.1 – 4.3.1 16 – 18 July 9, 2012 KitKat 4.4 – 4.4.4 19 – 20 October 31, 2013 Lollipop 5.0 – 5.1.1 21 – 22 November 12, 2014 Marshmallow 6.0 – 6.0.1 23 October 5, 2015 Nougat 7.0 24 August 22, 2016 Nougat 7.1.0 – 7.1.2 25 October 4, 2016 Oreo 8.0 26 August 21, 2017 Oreo 8.1 27 December 5, 2017 Pie 9.0 28 August 6, 2018 Android 10 (Q) 10.0 29 September 3, 2019 Android 11 (R) 11.0 30 September 8, 2020 Android 12, 12L (S) 12.0 31 – 32 October 4, 2021 Android 13 (TIRAMISU) 13.0 33 August 15, 2022 Android 14 (UPSIDE_DOWN_CAKE) 14.0 34 TBD (adsbygoogle = window.adsbygoogle || []).push({});