Migrating to API 26 (Android Oreo 8.0) for Android Enterprise

Action Required

Google Play (Including Play for Work) will require that new apps and app updates target at least API level 26 (Android Oreo 8.0) by November 2018. You must make this change or, you will not be able to update your app on the Play. You can find your current targetSdkVersion by looking in your gradle (if you use Manifest Entries, enabled by default) or AndroidManifest.xml files and while updating to a recent version is explicitly increasing that value, please see the code and configuration changes to make based on the sections below.

This guide has modifications for Android Enterprise, the original guide can be found here.

How to use this guide:
The older target API level your app has, the more effort will be typically required to update. This guide outlines the most important behavioral changes and deprecations that you will need to consider at each major API level in order to migrate to target API level 26 (or above). ***You start by reading the section for your current target API level and continue to the end of the guide, make sure to test with work profile compatibility.***

My App is used within Android Work Profile

- Set up and test managed profiles
    - A managed profile is controlled by an administrator, which may have limited functionality
        - Prevent failed intents, check for restrictions on if intents can resolve across profiles.
        - Sharing files across profiles, check to make sure to use content URIs.
        - Compatibility testing, ensure authentication, accounts, and more work properly.
- Using TestDPC to test Managed Configurations and other settings that your app handles.
    - For more information on how to set up TestDPC in various environments please see the github page.

- Make sure to use ANDROID_ID instead of any type of hardware identifier, it was deprecated in Marshmallow.

My App Targets pre-Android Lollipop (target API < 21)

- See "Feature Overview" pages for the following releases to ensure your app has accounted for changes introduced in these releases.
    - Lollipop
    - Jelly Bean
    - KitKat

My App Targets pre-Android Marshmallow (target API < 23)

- Runtime Permissions
    - Dangerous permissions are only granted at runtime. Adding runtime permissions will involve changes to certain UI flows.
    - Requesting access to these permissions needs to be afforded by your UI.
    - Handle rejection of permissions where possible (e.g. for location apps)
- *For an exhaustive list of changes introduced in Android Marshmallow, see feature overview.*

My App Targets pre-Android Nougat (target API < 24)

- Doze and App Standby
  You should design for behaviors described in "Optimizing for Doze and App Standby", which encompasses incremental changes introduced across a number of platform releases.
    - Triggered on device unplugged & screen off for time period
    - Network access is restricted
    - Alarms, Syncs, and Jobs are deferred
    - GPS & WiFi Scans are restricted
    - Normal priority FCM messages are restricted

- Background optimizations
    - *See Background Execution Limits (API 26) for restrictions on implicit broadcasts and background code execution. It encompasses incremental changes listed in this section.*
    - Removes 3 Implicit broadcasts CONNECTIVITY_ACTION, ACTION_NEW_PICTURE and ACTION_NEW_VIDEO but are superseded by the removal of most implicit broadcasts introduced in API 26.

- Permission Changes
    - Private directory of apps has restricted access
    - Exposing `file://` URI outside of your app triggers FileUriExposedException, developers who need to share files outside of their apps should implement FileProvider
- Linking to non-NDK libraries is blocked
- *For an exhaustive list of changes introduced in Android Nougat, see feature overview.*

My App Targets pre-Android Oreo (target API < 26)

- Background Execution Limits
    - Services are restricted when app not in foreground
        - startService() now throws exceptions when
        - Must use startForegroundService() and startForeground() for foreground services
        - See changes in JobScheduler background code execution
        - If FCM is used, Google Play services SDK should be version 10.2.1 or higher
        - FCM message delivery is subject to background execution limits. When background work is necessary upon message receipt (for example, perform background data sync), schedule jobs through FJD job or JobIntentService. See FCM messages documentation for detail.
    - Implicit broadcasts are restricted
        - See JobScheduler for handling background events
- Background Location Limits
    - Background apps have limited access to location data
        - Active / Foreground apps are unaffected
        - On devices with Google Play services, use Fused Location provider to get periodic location updates
- Notification Channels
    - Notifications interruption properties should be moved to a channel
    - All Notifications should be posted with a channel otherwise they won't be notified
    - Consider using NotificationCompat.Builder for handling backwards compatibility of notification channels
- Privacy
    - AndroidID is scoped per app signing key
- *For an exhaustive list of changes introduced in Android Oreo, see feature overview.*

Modernizing your apps

As you update the target API level for your apps, consider adopting recent platform features to modernize your apps and delight your users.

- Upgrade to the latest version of FCM
    - Migrate GCM client app for Android to FCM
- Advanced window management
    - Support larger aspect ratio (more than 16:9) to take advantage of recent device models. Ensure your app resizes to fill the available screen space. Declare a maximum aspect ratio as a last resort.
    - Add Multi-window support to increase productivity with your app
    - Add Multi-display support for devices with multiple displays
    - Support Picture-in-Picture for a great minimized app experience for appropriate use cases.
    - Optimize for devices with display cutout
        - Do not assume status bar height. Use WindowInsets and View.OnApplyWindowInsetsListener (video explanation)
        - Do not assume the app has the entire window. Use View.getLocationInWindow(), not .getLocationOnScreen()
        - When handling MotionEvent, use MotionEvent.getX(), .getY(), not MotionEvent.getRawX(), .getRawY()
- Modern camera support
    - Use Camera2 API to make the most of using the camera
    - Enable Pixel Visual Core to accelerate HDR+ processing on Pixel 2 devices

Check and update your SDKs and libraries

Make sure that your third-party SDK dependencies support API 26 - some SDK providers publish it in their manifest, others will require additional investigation. If you use an SDK that doesn't support API 26, please make it a priority to work with the provider to resolve the issue.

To investigate whether an SDK dependency supports API 26, the same checklist above would apply.

Additionally note that your app or game's targetSdkVersion may restrict access to private Android platform libraries, see NDK Apps Linking to Platform Libraries for details.

Verify your Android Support Library

As always, you must ensure compatibility between the major version of Android Support Library and your app's `compileSdkVersion`.

We recommend the choice of `targetSdkVersion` is smaller or equal to the Support Library's major version – you are encouraged to update to a recent compatible Support Library in order to take advantage of the latest compatibility features and bug fixes.

Test your app

Once your app is targeting API 26 and handled all the highlighted features above, you should test core use cases. The following information aims to guide your testing process and is by no means exhaustive, which is why we advise following the above guides to adopt features rather than updating the targetSdkVersion and fix errors. We suggest testing the following:

- Compiles to API 26 without errors
    - Fix any compilation warnings such as API deprecations and errors
- Handles disabling permissions and prompts user if required
    - In the App info test disabling each permission
    - Open the app and ensure no crashes
    - Perform core use case tests and ensure required permissions are re-prompted
- Handles Doze with expected results and no errors
    - Set your app into the Doze states
        - Test any use cases that trigger FCM messages
        - Test any use cases that use Alarms or Jobs
        - *At this point, you shouldn't be relying on background services*
    - Set your app into App Standby
        - Test any use cases that trigger FCM messages
        - Test any use cases that use Alarms
- Handles new photos / video being taken
    - Checking your app handles the NEW_PICTURE and NEW_VIDEO changes correctly (i.e. moved to JobScheduler jobs)
    - Ensure that any critical use cases that depend on these events still work
- Handles sharing files to other apps
    - Test any use case that shares file data with any other app (especially another another of your own apps)
        - Test the content is visible in the other app and doesn't trigger crashes

---

Behavior Changes by Theme

To figure out the impact of the upgrade by android theme, use the following table:

If you use:	**You need to worry about:**
Work Profile	Managed Profiles, Managed Configuration
Permissions	Runtime Permissions
Background Execution	Doze Doze Light Background Optimization Background Execution Limits
Notifications	Notification Channels
Foreground service	startForegroundService and startForeground
Location	Background Execution Limits Background Location Limits
Camera	Background Optimization Runtime Permissions
NDK	64 bit ABI support