DevCon 2025

Connect | Learn | Build

June 4 - 6, 2025

Amsterdam, Netherlands

ZEBRA

DevCon 2025

Connect | Learn | Build

Harness What's New in Android for Optimal Development

Rajeev Varma
Sr Manager, Software Engineering

Nicola De Zolt L.
Software Engineer

ZEBRA

Agenda

Harness What's New in Android for Optimal Development

DevCon 2025
Connect | Learn | Build

Security

- Android 14 and 15 security and privacy features: Minimum Installable API, Credential Manager, Screenshot and Screen recording detection, Partial screen sharing, E2E Encryption, Overlay restrictions

Zebra MX, Android Core And Performance Features

- Zebra Mx
- Exact Alarms, Typed FGS, Broadcast Receivers, Loudness controls, Large screen improvements

Migrating, Debugging, Optimizing

- Reduce your anxiety when migrating
- Availability of Android 14 & 15 for Zebra devices
- Memory Optimization & Power Management Guidelines

DevCon 2025

Connect | Learn | Build

Security

Android Security & Privacy Features

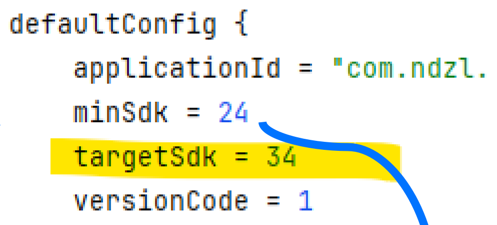
Impact categories - Legend

DevCon 2025
Connect | Learn | Build

Each new OS feature might impact apps in three different ways

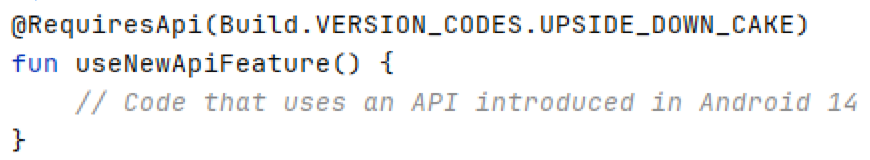
Impacts all APPs running on Android regardless of the app's targetSdkVersion.

Impacts APPs targeting Api Level 3X+

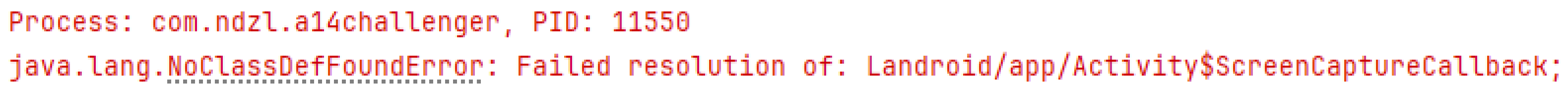


```
defaultConfig {
    applicationId = "com.ndzl.
    minSdk = 24
    targetSdk = 34
    versionCode = 1
```

New feature — **New feature!**



```
@RequiresApi(Build.VERSION_CODES.UPSIDE_DOWN_CAKE)
fun useNewApiFeature() {
    // Code that uses an API introduced in Android 14
}
```



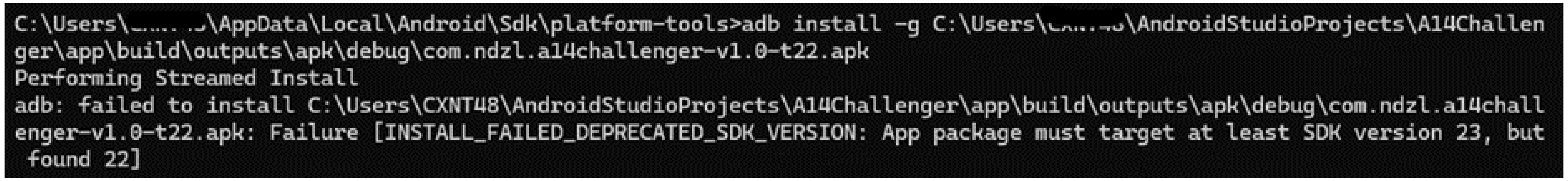
```
Process: com.ndzl.a14challenger, PID: 11550
java.lang.NoClassDefFoundError: Failed resolution of: Landroid/app/Activity$ScreenCaptureCallback;
```

Minimum Installable Target API

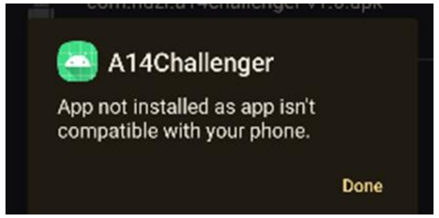
Impacts all APPs running on Android 14/15

DevCon 2025
Connect | Learn | Build

Minimum installable target API level is 23 for Android 14 and API level 24 for Android 15
https://developer.android.com/about/versions/14/behavior-changes-all#minimum-target-api-level

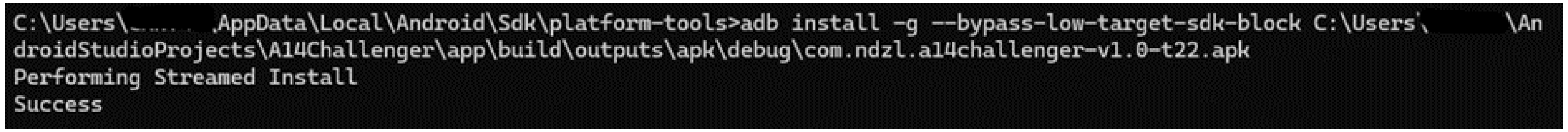


```
C:\Users\[redacted]\AppData\Local\Android\Sdk\platform-tools>adb install -g C:\Users\[redacted]\AndroidStudioProjects\A14Challen
ger\app\build\outputs\apk\debug\com.ndzl.a14challenger-v1.0-t22.apk
Performing Streamed Install
adb: failed to install C:\Users\CXNT48\AndroidStudioProjects\A14Challenger\app\build\outputs\apk\debug\com.ndzl.a14chall
enger-v1.0-t22.apk: Failure [INSTALL_FAILED_DEPRECATED_SDK_VERSION: App package must target at least SDK version 23, but
 found 22]
```



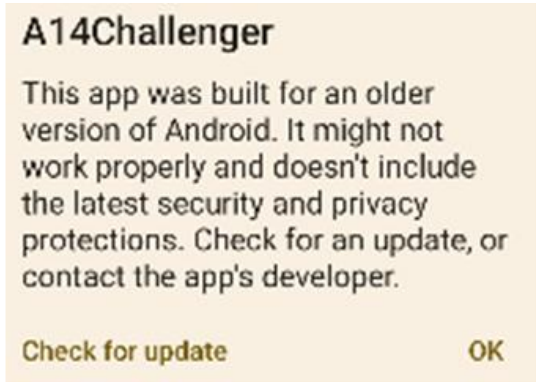
A14Challenger
App not installed as app isn't compatible with your phone.
Done

- Bypassing the restriction

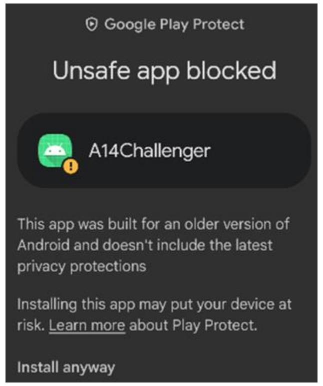


```
C:\Users\[redacted]AppData\Local\Android\Sdk\platform-tools>adb install -g --bypass-low-target-sdk-block C:\Users\[redacted]\An
droidStudioProjects\A14Challenger\app\build\outputs\apk\debug\com.ndzl.a14challenger-v1.0-t22.apk
Performing Streamed Install
Success
```

- Prompts



A14Challenger
This app was built for an older version of Android. It might not work properly and doesn't include the latest security and privacy protections. Check for an update, or contact the app's developer.
Check for update OK



Google Play Protect
Unsafe app blocked
A14Challenger
This app was built for an older version of Android and doesn't include the latest privacy protections
Installing this app may put your device at risk. Learn more about Play Protect.
Install anyway

Credential Manager

New feature in Android 14

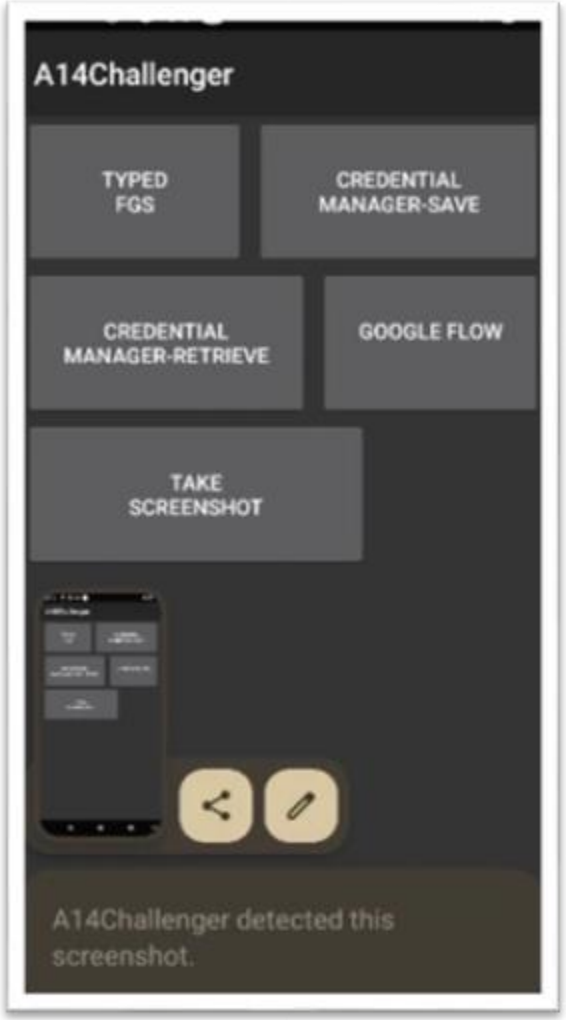
DevCon 2025
Connect | Learn | Build

- A unified system that simplifies how users sign in to apps and websites, supporting multiple sign-in methods
- Backward compatible with Android 4.4 (API level 19) through a Jetpack Library using Google Play services
- Credential Manager includes authentication workflows like:
  - Traditional **username and password**
  - **Enhanced Autofill:** a feature that provides secure, streamlined storage and retrieval of username/password pairs; leverages Android credential provider
  - **Passkeys** - based on the Fido Alliance industry-standard https://fidoalliance.org/how-fido-works/ and https://developer.android.com/identity/sign-in/credential-manager#about-passkeys – Passkeys can be synchronized across devices in the same ecosystem.
    - Also, developers using FIDO2 protocol are advised to migrate to Credential Manager https://developer.android.com/identity/sign-in/fido2-migration
  - **Federated sign-in solutions:** allows users to authenticate using their existing accounts from identity providers like Google, Apple, or other OAuth providers.
  - Consistent user experience

Screenshot Detection

New feature in Android 14

- Aims to prevent apps from indiscriminately tracking user activity (potentially compromising sensitive information captured in screenshots)
- A new API, **Activity.ScreenCaptureCallback**, allows apps to detect screenshots only within their app's bounds (DETECT_SCREEN_CAPTURE permission)
- Only *manually* taken screenshots are detected! And only within the specific Activity. (Power button + Volume Down)
- Note that starting with Mx5.0+, Zebra provided a way to block screenshots. Refer to Techdocs/#screen-shot-enabledisable



A14Challenger detected this screenshot.

Screenshot Detection

New feature

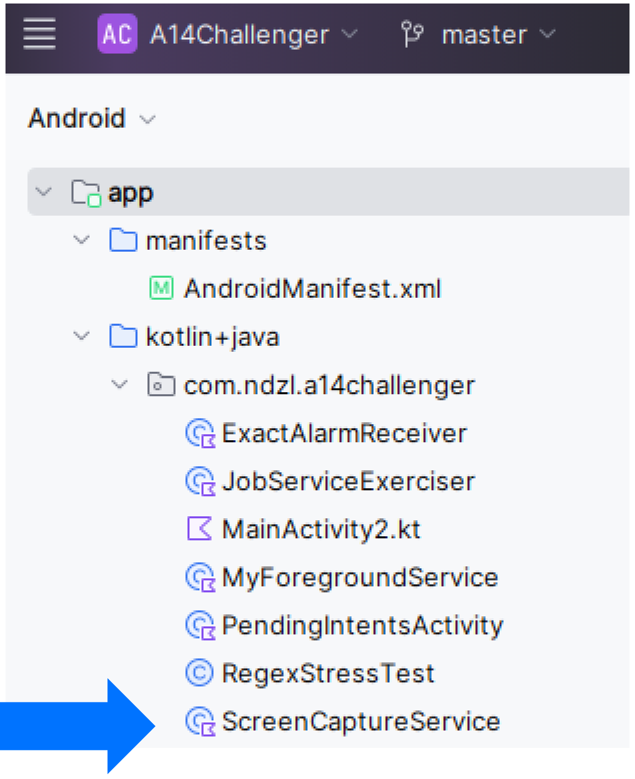
Important note:

- Activity. ScreenCaptureCallback **does not capture** screenshots taken through the *Media Projection APIs*!
- Media Projection is based on a Service (of Type *mediaProjection*)
- Refer to the ScreenCaptureService in my A14Challenger sample code on ZebraDevs Github organization.

To block any screenshots
(manual and through MediaProjection) use this call

```
activity.getWindow().setFlags(LayoutParams.FLAG_SECURE, LayoutParams.FLAG_SECURE)
```

A14 Blog link



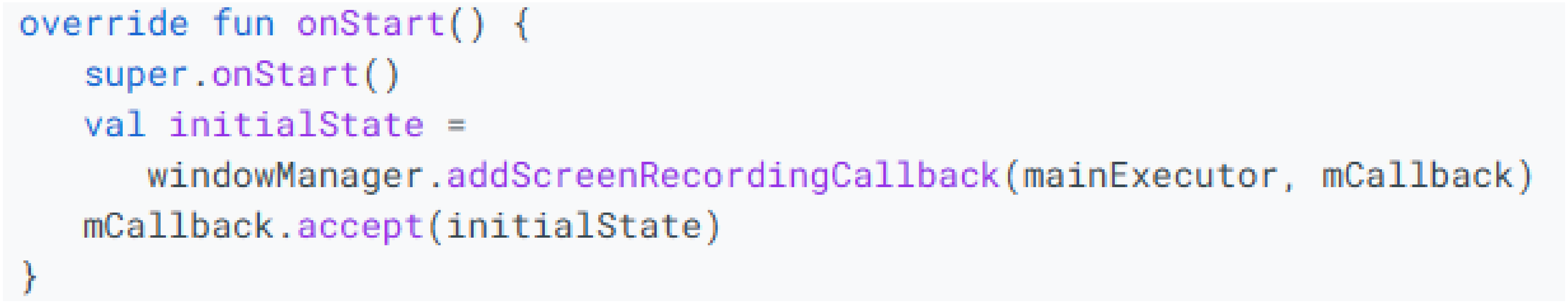
Screen recording detection

New feature

DevCon 2025
Connect | Learn | Build

A callback is invoked whenever the app transitions between being visible or invisible within a screen recording. An app is considered visible if activities owned by the registering process's UID are being recorded.

Reference
https://developer.android.com/about/versions/15/features#screen-recording-detection



```
override fun onStart() {
    super.onStart()
    val initialState =
        windowManager.addScreenRecordingCallback(mainExecutor, mCallback)
    mCallback.accept(initialState)
}
```

Partial screen sharing

New feature

DevCon 2025
Connect | Learn | Build

Users can share or record **just an app** window rather than the entire device screen. This feature includes MediaProjection **callbacks** that allow your app to customize the partial screen sharing experience. This is relevant in the **privacy** context, enhances multitasking by enabling users to run multiple apps but restrict content sharing to a single app.

The newly added MediaProjection APIs are:

- **onCapturedContentResize()**
  Enables resizing of the shared projection based on the size of the captured display area.
- **onCapturedContentVisibilityChanged()**
  Informs the shared projection host app of the visibility of the capture content

Reference
https://developer.android.com/about/versions/14/features/app-screen-sharing

Key management for end-to-end encryption

New feature

DevCon 2025
Connect | Learn | Build

Android 15 is introducing the E2eeContactKeysManager in Android 15, which facilitates end-to-end encryption (E2EE) in your Android apps by providing an OS-level API for the **storage of cryptographic public keys.**

The E2eeContactKeysManager is designed to give users a centralized way to manage and **verify their contacts' public keys.**

Keys are uniquely identified by:

- **ownerPackageName** - package name of an app that the key belongs to.
- **deviceId** - an app-specified identifier for the device. The deviceID should be unique among devices belonging to that user.
- **accountId** - for most apps this would be a phone number.

E2eeContactKeysManager manages two types of keys: E2eeContactKey (a public key associated with a contact for user-contact communication) and E2eeSelfKey (a key that represents the owner of the device)

Reference
https://developer.android.com/about/versions/15/features#key-mgmt-e2ee

Overlay restrictions

New feature for Zebra Android 13+

DevCon 2025
Connect | Learn | Build

getWindow().setHideOverlayWindows(true);

- Prevents **non-system overlay windows** from being drawn on top of the window it is called from
- Requires Manifest.permission.HIDE_OVERLAY_WINDOWS
- For a coding example, refer to this sample (or frame the Qrcode)

DevCon 2025

Connect | Learn | Build

Zebra MX, Android Core Features, and Performance

ZEBRA

MX

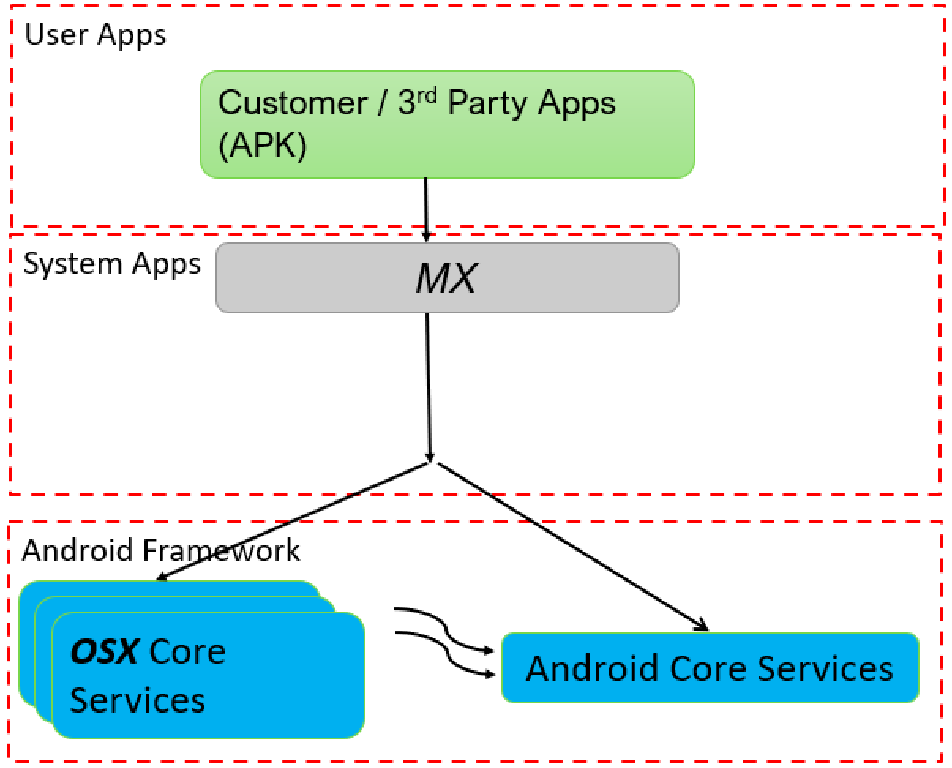
A closer look at the Zebra value add

- Mx is an abstraction layer for Application developers
- As of Today, Mx provides access to ***300+*** Enterprise Grade APIs.

MX FEATURES

- Application Lockout
- Certificate Management
- Silent Application Management (Installation/Uninstallation)
- Secure Storage
  - Encrypt Phone Data & SD card
- Default Home Screen
- Privileged Command Runner
- Event Injection Service
- Peripheral Device Management
  - Bluetooth
  - NFC
  - USB
  - Camera
  - Imager
- Threat Counter Measure Service
- … and more

- Developers will always interface to well defined CSP's (Configuration Service providers) that don't change,



Exact Alarms

Impacts all APPs running on Android 14+

DevCon 2025
Connect | Learn | Build

From Android 12

- It will no longer be possible to freely set & schedule exact alarms through the AlarmService class
- APPs willing to preserve this feature will have to get access to the "Alarms & Reminders" capability that appears within the Special App Access Screen in System Settings.
- The special app access is obtainable by including this permission in the AndroidManifest: **SCHEDULE_EXACT_ALARM**
- Pre-granted by default only on this version

From Android 13

- The **USE_EXACT_ALARM** permission has been introduced
- Granted automatically
- Cannot be revoked by the user
- Subject to the upcoming Google Play's policy
- Limited use cases (only if a user-facing function in your app requires precisely-timed actions)
- **SCHEDULE_EXACT_ALARM** permission is no longer granted by default and developer needs to check if the user has granted it

From Android 14

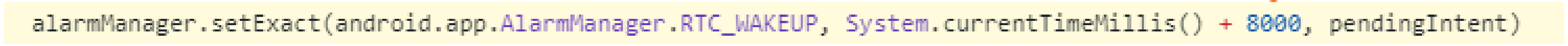
- **SCHEDULE_EXACT_ALARM** is now denied by default, so developers will need to ask this permission accordingly
- If user backup and restore the application on a different device preserving data, the permission will still be denied
- If the application already has the permission granted by the user coming from an earlier Android version, it will be persisted on Android 14 as well

Exact Alarms

Impacts all APPs running on Android 14+

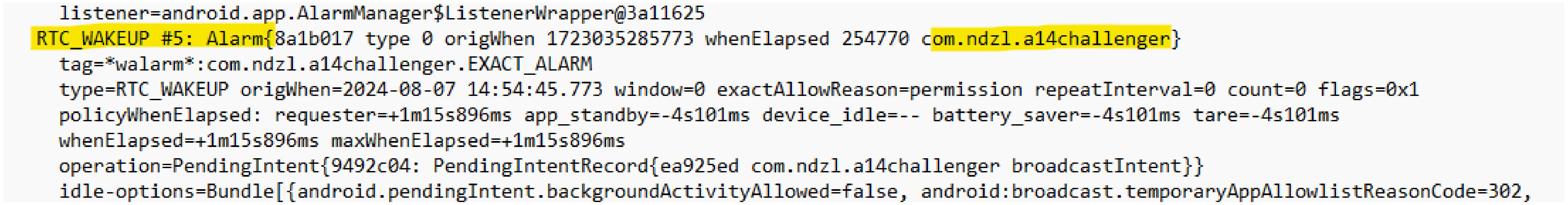
DevCon 2025
Connect | Learn | Build

- Here is how to schedule an Alarm



```
alarmManager.setExact(android.app.AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + 8000, pendingIntent)
```

- Scheduled alarms are system resources – use **adb shell dumpsys alarm** to list them



```
    listener=android.app.AlarmManager$ListenerWrapper@3a11625
  RTC_WAKEUP #5: Alarm{8a1b017 type 0 origWhen 1723035285773 whenElapsed 254770 com.ndzl.a14challenger}
    tag=*walarm*:com.ndzl.a14challenger.EXACT_ALARM
    type=RTC_WAKEUP origWhen=2024-08-07 14:54:45.773 window=0 exactAllowReason=permission repeatInterval=0 count=0 flags=0x1
    policyWhenElapsed: requester=+1m15s896ms app_standby=-4s101ms device_idle=-- battery_saver=-4s101ms tare=-4s101ms
    whenElapsed=+1m15s896ms maxWhenElapsed=+1m15s896ms
    operation=PendingIntent{9492c04: PendingIntentRecord{ea925ed com.ndzl.a14challenger broadcastIntent}}
    idle-options=Bundle[{android.pendingIntent.backgroundActivityAllowed=false, android:broadcast.temporaryAppAllowlistReasonCode=302,
```

Typed Foreground Services

Impacts APPs targeting Api Level 34+

- **Foreground services** are used for tasks that need to be immediately noticeable to the user, such as ongoing notifications or critical operations.
- Android 14 requires apps to declare specific foreground service *types* to improve clarity and control over foreground services.
- By requiring apps to specify the type of foreground service, Android 14 ensures that these services are used appropriately and that users better understand why a service is running.
- When a use case in your app isn't associated with any of these types, Google recommends that you migrate your logic to use WorkManager or user-initiated data transfer jobs

```
<service
    android:name=".MyForegroundService"
    android:enabled="true"
    android:exported="false"
    android:foregroundServiceType="mediaPlayback" />
```

- `camera`
- `connectedDevice`
- `dataSync`
- `health`
- `location`
- `mediaPlayback`
- `mediaProjection`
- `microphone`
- `phoneCall`
- `remoteMessaging`
- `shortService`
- `specialUse`
- `systemExempted`

Runtime broadcast receivers export behavior

Impacts APPs targeting Api Level 34+

DevCon 2025
Connect | Learn | Build

- API Level 33 added the export specification as optional
- **Context-registered receivers** are required to specify a flag to indicate whether or not the receiver should be exported to all other apps on the device: either RECEIVER_EXPORTED or RECEIVER_NOT_EXPORTED
- Not required when receiving only *system* broadcasts

Restrictions on BOOT_COMPLETED broadcast receivers

Impacts APPs targeting Api Level 35+

- API Level 35 added the following restriction
  - Broadcast receivers launching foreground services. BOOT_COMPLETED receivers are not allowed to launch the following types of foreground services:
    - dataSync
    - camera
    - mediaPlayback
    - phoneCall
    - mediaProjection
    - microphone (since Android 14)
  - If a BOOT_COMPLETED receiver tries to launch any of those types of foreground services, the system throws *ForegroundServiceStartNotAllowedException*.

Loudness control

New feature (API 35)

DevCon 2025
Connect | Learn | Build

- Android 15 introduces support for **loudness standardization** for a better user experience
- This aims at avoiding audio loudness inconsistencies
- Ensure that users do not have to constantly adjust volume when switching between content
- LoudnessCodecController is a new class for getting recommended loudness parameter updates for audio decoders based on the current encoding and audio routing.
  - They ensure the loudness and dynamic range of the content is optimized to the physical characteristics of the audio output device (e.g. phone microspeakers vs headphones vs TV speakers).
  - Those updates can be automatically applied to the MediaCodec instance(s)

Improved large screen multitasking

New features added in API 35

DevCon 2025
Connect | Learn | Build

Android 15 gives users better ways to **multitask on large screen** devices.

For example, users can

- save their favorite split-screen app combinations for quick access
- pin the taskbar on screen to quickly switch between apps.

For a crash course on large-screen APIs, refer to my past presentation on Zebra Workstation Connect and UI design considerations.

DevCon 2025

Connect | Learn | Build

Migrating, Debugging, Optimizing

ZEBRA

Reduce Your Anxiety in Moving to the next Android Dessert

DevCon 2025
Connect | Learn | Build

In this blog, we highlighted how important it is to move to Android 14

- https://www.zebra.com/gb/en/blog/posts/2024/do-these-things-to-reduce-android-migration-anxiety.html (or frame the QrCode in this page)

Key Points:

- What needs to happen to get your mobile devices and applications up and running on Android 14.
- The best way for the migration to happen.
- The real reasons why this migration needs to happen now, even if you'd prefer to wait as long as possible.

What Android 14 is Going to Do for You/Your Operations

- **Latest security patches** and bug fixes, updates via the Google Play Store
- Latest third-party **applications and libraries**
- **New features** only available on the Android 14 OS

Get your Team Up and Running on Android 14

- **Learn** about all the changes that have been made in the new OS version you are targeting
- Confirm how they may affect your application
- Commit to upgrading to new OS versions and have a plan to periodically update

To read it thoroughly,
Refer to the link above or point your camera here >>>

Availability of Android 14 & 15 for Zebra devices

As of May 14, 2025

DevCon 2025
Connect | Learn | Build

Android 14 is currently available for the following platforms

- **QC SD660** (MC3300ax, MC20, RZ-H271, CC600, CC6000, EC30, EC50, EC55, ET51,ET56, L10A, MC2200, MC2700, MC3300x, MC3300xR, MC9300, TC21, TC21 HC, TC26, TC26 HC, TC52,TC52 HC, TC52x, TC52x HC, TC52AX, TC52AX HC, TC57, TC57x, TC72, TC77, TC8300, VC8300 & WT6300)

- **QC 6490/QC 5430** (TC22, TC27, TC53, TC58, TC73,TC78, ET60, ET65, HC20, HC25, HC50, HC55, EM45)

- **QC 6375** (ET40/45, TC15)

- **QC 4490** (MC34, MC94, TC53e/58e, WT54/64, PS30)

Android 15
- To be released by Q3/Q4 2025

Memory optimization

Guidelines

DevCon 2025
Connect | Learn | Build

Understand the Android Memory Model
Android apps run in their own process and have a limited amount of memory available. Exceeding this limit can lead to crashes (OutOfMemoryError).
Be aware of the lifecycle of activities and fragments, and manage their memory usage accordingly.

Minimize Memory Usage
Avoid memory leaks by ensuring that references to activities or views are not held longer than necessary.
Use weak references where applicable, especially for callbacks and listeners.

Efficient Resource Management
Use efficient data structures; prefer arrays or ArrayLists over LinkedLists unless there's a specific need.
Reuse bitmaps and other heavy resources when possible. Use BitmapFactory.Options for efficient image loading.
Consider using the Glide or Picasso libraries for image loading and caching, as they handle many memory management tasks automatically.

Manage Activity and Fragment Lifecycles
Release resources such as bitmaps, cursors, and other large objects in onPause() or onStop().
Cancel ongoing tasks and threads when an activity or fragment is stopped.

Use Memory Profiler Tools
Utilize Android Studio's Memory Profiler to monitor memory usage, detect leaks, and understand the app's memory footprint.

Optimize Data Handling
Avoid loading large data sets into memory all at once. Use paging or lazy loading techniques.
Use SQLite, Room, or other database solutions for large datasets instead of keeping them in memory.

Garbage Collection
Be aware of the garbage collector's behavior. Avoid creating unnecessary objects and consider object pooling for frequently used objects.

Background Services
Limit the use of background services. Prefer WorkManager, JobScheduler, or Firebase Cloud Messaging for background tasks.

Optimize UI Components
Use ViewHolder pattern in RecyclerViews to minimize unnecessary view inflation.
Avoid complex view hierarchies and deep nesting in layouts.

Consider Low-Memory Devices
Test your app on a variety of devices, including those with lower memory, to ensure it behaves well across the board.

Power Management

Guidelines

DevCon 2025
Connect | Learn | Build

- Apps should use foreground services when absolutely necessary and stop them as soon as possible
- For background job execution, use WorkManager, which is efficient and respects battery optimization settings instead of threads and background services
- Minimize the use of wake locks and release them as soon as your task is complete
- Leverage the Doze mode of Android. Respect the Doze mode and App Standby by scheduling jobs with JobScheduler, AlarmManager, or WorkManager. This ensures job execution during maintenance windows
- Limit the use of broadcast receivers and unregister them when they are not needed
- Do not ask for permissions that are not required for an app functionality
- Choose efficient data formats (e.g., JSON instead of XML) to reduce the amount of data transmitted
- Do not use files for IPC. Use standard Android constructs like binders, intents, etc.
- Limit the depth of nested activities in app design
- Request location updates only when necessary and at the lowest frequency needed for your app

Online references

DevCon 2025
Connect | Learn | Build

Planning A14 migration blog
https://www.zebra.com/gb/en/blog/posts/2024/do-these-things-to-reduce-android-migration-anxiety.html

Android 14 official doc
https://developer.android.com/about/versions/14/summary

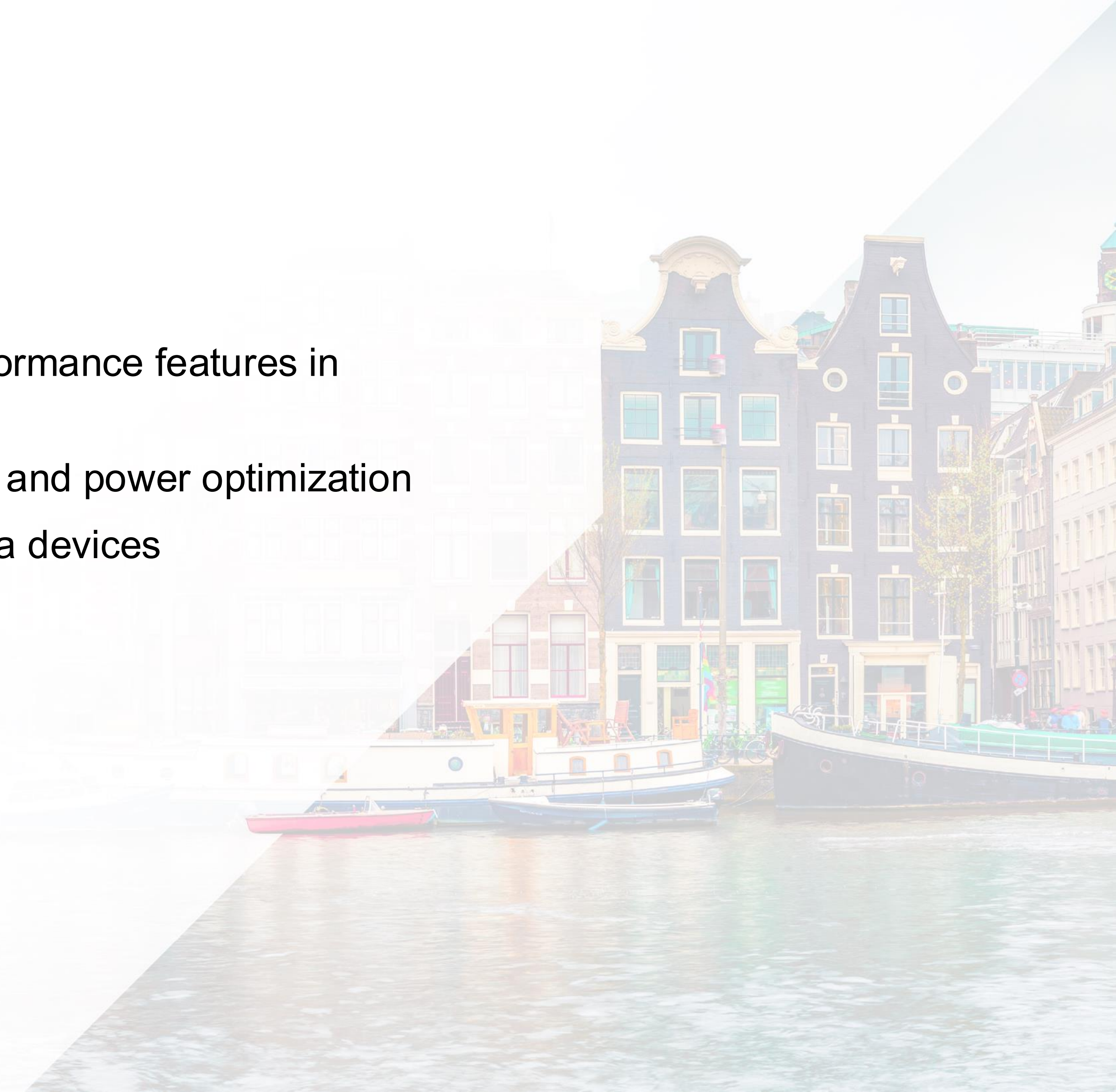
What's New in Android™ 14 and the Impact on Zebra developers
https://developer.zebra.com/blog/whats-new-androidtm-14-and-impact-zebra-developers

Android 15 official doc
https://developer.android.com/about/versions/15/summary

Summary

Today we covered these topics

- Security features in Android 14/15
- MX and the most relevant Core and Performance features in Android 14/15
- Guidelines to support migration, memory and power optimization
- Availability of Android 14 and 15 for Zebra devices



DevCon 2025

Connect | Learn | Build

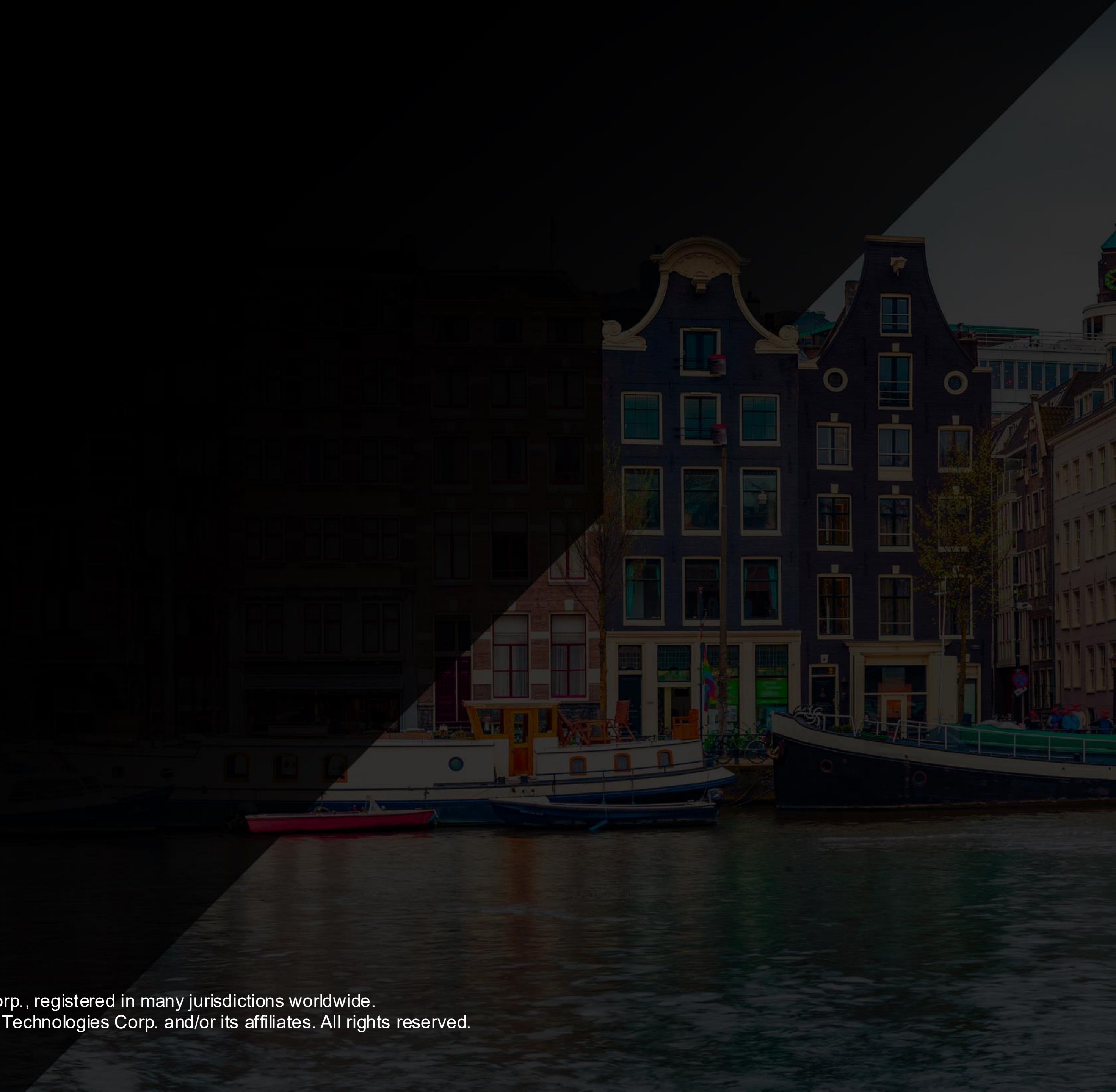
Questions?

DevCon 2025

Connect | Learn | Build

Thank You

ZEBRA



Private space

New feature

Private Space is a feature that creates **a separate, secure environment** within a device.

- **Separate Profile**
  - Creates an *isolated user profile*
  - Has its own apps, data, and settings
  - Cannot be accessed from the main profile
- **Key Features**
  - Independent storage
  - Custom notification settings
  - No cross-profile data sharing
- **Security**
  - Protected by different authentication than the main profile
  - No screenshots allowed by default

Installing an APK only for the Private Space:

- Similar to a work profile
- Identify the ***userID*** linked to the private space profile with
  dumpsys user | grep "Private space"
- Call
  adb install --***userID*** your-app.apk

PDF Renderer

New features added in API 35

DevCon 2025
Connect | Learn | Build

PdfRenderer already existed since API21

In API35 new features have been added. E.g.

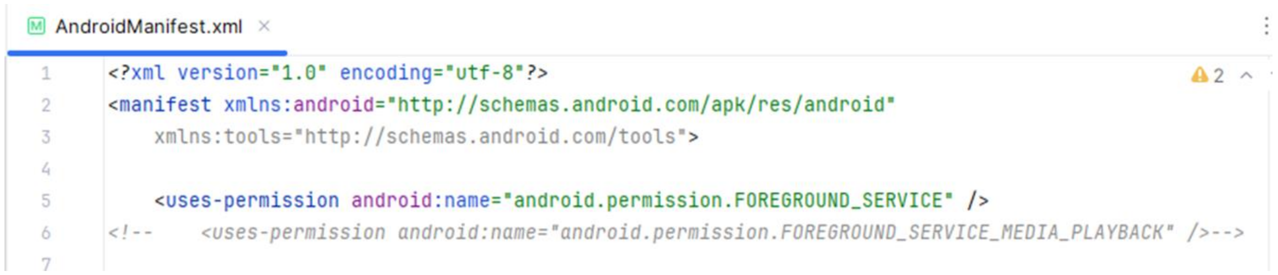
- Rendering password-protected files
- Annotations
- Form editing
- Searching, and selection with copy.

Debugging tools

Compatibility Framework – Supporting OS Migration

DevCon 2025
Connect | Learn | Build

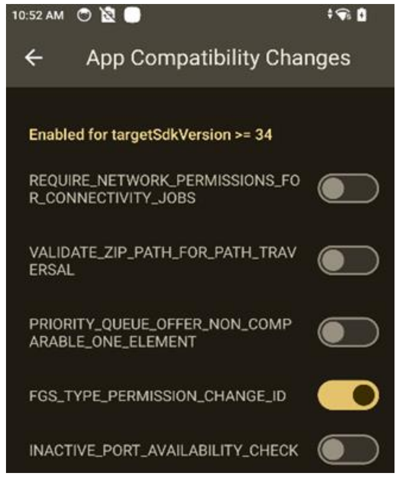
- Helps to identify compatibility changes impacting an app
- E.g. FGS Types - App targeting API 29 (A10), running on A14 => No issues



AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<!--    <uses-permission android:name="android.permission.FOREGROUND_SERVICE_MEDIA_PLAYBACK" />-->
```

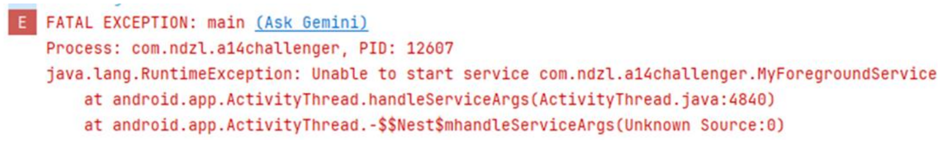
- Now emulate API 34  and you'll get a runtime exception

10:52 AM

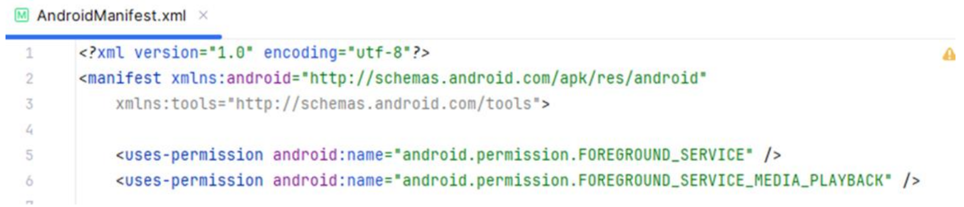
App Compatibility Changes

Enabled for targetSdkVersion >= 34

- REQUIRE_NETWORK_PERMISSIONS_FOR_CONNECTIVITY_JOBS
- VALIDATE_ZIP_PATH_FOR_PATH_TRAVERSAL
- PRIORITY_QUEUE_OFFER_NON_COMPARABLE_ONE_ELEMENT
- FGS_TYPE_PERMISSION_CHANGE_ID ✓
- INACTIVE_PORT_AVAILABILITY_CHECK



```
E  FATAL EXCEPTION: main (Ask Gemini)
   Process: com.ndzl.a14challenger, PID: 12607
   java.lang.RuntimeException: Unable to start service com.ndzl.a14challenger.MyForegroundService
       at android.app.ActivityThread.handleServiceArgs(ActivityThread.java:4840)
       at android.app.ActivityThread.-$$Nest$mhandleServiceArgs(Unknown Source:0)
```



AndroidManifest.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
    <uses-permission android:name="android.permission.FOREGROUND_SERVICE_MEDIA_PLAYBACK" />
```

Migration Roadmap

Checklists to run on Android 14/15

DevCon 2025
Connect | Learn | Build

CASE #1 – Your App is currently targeting API 33/34 and running on Android 13/14 respectively

- Review your code and check for gaps against the changes affecting all apps
  - You need to clear all the requirements
- Optionally upgrade your targetSDK level to 34/35
  - The Compatibility Framework helps you step-by-step identifying potential gaps
- Optionally leverage the new Android 14/15 features

CASE #2 – Your App is currently targeting API 29 and running on Android 11

- Take the chance to align the targetSDK level to 34/35
- Fill all the gaps in terms of requirements to run on Android 14/15
  - Work on requirements from Android 14/15 backward
- Optionally leverage the new Android 14/15 features