Secure Capture — Real-Time, On-Device, No Storage

Purpose
Analyze frames from the front-facing camera entirely **on device** to enhance fraud detection—**no images are stored or transmitted; only metadata/results are sent**.

How it works

- App decides when to activate (e.g., after login, during a transfer, or only in high-risk flows).
- `startCapture()` checks camera permission and begins periodic frame analysis locally.
- Frames are processed on device; images are **deleted immediately** after analysis.
- Only analysis results/metadata are sent to your backend.
- Capture ends when the app calls stopCapture().

Required permission (AndroidManifest.xml)

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

SDK methods & returns

- `startCapture()` → `SUCCESS` | `MISSING_PERMISSIONS`
- `stopCapture()` → `SUCCESS`

Usage example (Kotlin)

```
val result = collectorAgent.startCapture()
if (result == "MISSING_PERMISSIONS") {
    // Request CAMERA permission before retrying
}
// ... perform secured flow ...
collectorAgent.stopCapture()
```

UI guidance (pre-permission message)}

> “To enhance security, the app will analyze images from your device’s front-facing camera. No images are stored or uploaded.”

Security & privacy notes

- No images ever leave the device; only minimal metadata/results are transmitted.
- Immediate buffer deletion after analysis; follow data minimization best practices.

Common scenarios

- Post-login environment validation
- Extra check during standard transfers
- High-risk events (new payee, unusually large amount)

Next steps

1. Add CAMERA permission.
2. Explain purpose to users.
3. Integrate startCapture()/stopCapture() in the chosen flow(s).
4. Handle permission gracefully. 5) Roll out gradually and monitor analytics.

Revision #4
Created 18 September 2025 15:48:03 by roger de avila
Updated 18 September 2025 16:05:59 by roger de avila