Technische Dokumentation

**Fehlerbehebung, Reanimation und NDEF-Codierung des ACS ACR1555U-A1 unter Android/Flutter**

**Projekt:** druckpunkt (Atemschutzüberwachung)

**Autor:** Blaulicht Schmiede

**Relevanz:** Für den Betrieb von externen NFC-Lesern.

---

**TEIL 1: Leitfaden für Anwender & Administratoren (EndUser-Ebene)**

**1.1 Das Problem: Der "Tiefschlaf-Lockout"**

Im laufenden Betrieb mit der Atemschutz-App konnte es passieren, dass der Reader nach dem Trennen der Bluetooth-Verbindung oder nach Ablauf des automatischen Energiespar-Timers mit einem **Doppelpiep** ausging. Danach war das Gerät wie „tot“: Der Power-Button reagierte nicht mehr, und der Reader ließ sich auch per Ladekabel weder einschalten noch ansprechen.

**1.2 Die Sofortmaßnahme im Depot: Der Hardware-Kaltstart**

Wenn der Reader in diesem Zustand feststeckt, hilft keine Tastenkombination mehr. Das Gerät hat sich intern elektronisch komplett verriegelt. Um den Reader sofort wiederzubeleben, muss er einmalig physisch stromlos gemacht werden:

1. **Gehäuse öffnen:** Die Gehäuseschalen des ACR1555U vorsichtig trennen.
2. **Akku abziehen:** Das kleine Akku-Kabel (drei Adern, drei Farben) vorsichtig von der Hauptplatine abziehen.
3. **Reststrom leeren:** Den Power-Button bei abgezogenem Akku für ca. 10 Sekunden gedrückt halten, um die Kondensatoren auf der Platine restlos zu entladen.
4. **Wieder anstecken:** Nach mind. 2 Minuten Wartezeit den Akku wieder aufstecken. Das Gerät startet sofort neu und ist wieder einsatzbereit.

**1.3 Die dauerhafte Lösung (Firmware-Update)**

Die Ursache für diesen Absturz ist ein Fehler in der ab Werk ausgelieferten Firmware-Linie des Herstellers ACS. Für einen fehlerfreien und unzerstörbaren Betrieb im Atemschutzeinsatz muss das Gerät zwingend auf die **Firmware-Version 1.03.02** (oder neuer) aktualisiert werden.

Da diese Dateien dem Urheberrecht unterliegen, dürfen sie nicht frei im Netz verteilt werden. Wir haben daher am Ende dieses Artikels eine fertige E-Mail-Vorlage vorbereitet, mit der du das Update-Paket kostenlos direkt beim Hersteller anfordern kannst.

**TEIL 2: Technischer Deep Dive (Entwickler- & System-Ebene)**

### 2.1 Die detaillierte technische Problemstellung

Bei der tieferen Analyse der Log-Dateien und Protokolle während der Entwicklung unserer Flutter-App stießen wir auf drei Kernprobleme im Zusammenspiel zwischen der Android-SmartCardIO-Architektur und dem ACS-Mikrocontroller:

1. **Der Kernel-Panic-Freeze des PMIC:** Auf den alten Firmware-Versionen (z. B. 1.02.08) war der Reader standardmäßig im *BLE HID Modus* (Tastatur-Emulation) konfiguriert. Riss die Verbindung ab, versuchte der Controller, ein Bluetooth-Abmeldesignal als HID-Gerät zu senden. Dies führte zu einem harten Registerkonflikt im EEPROM und einer *Kernel Panic* des internen Betriebssystems. Da der Akku dauerhaft Spannung lieferte, blieb dieser Absturz-Zustand im flüchtigen SRAM eingefroren und blockierte die Abfrage des Power-Buttons.
2. **Asynchroner GATT-Wettlauf (Race Condition):** Unsere reaktive Lese-Schleife pollt das Antennenfeld im 250ms-Raster mittels nativer connect("*")-Aufrufe. Wurde nun aus der Flutter-Oberfläche heraus ein Schreibbefehl abgefeuert, kollidierten Lese- und Schreibzugriffe auf der Bluetooth-GATT-Pipeline. Das Resultat waren JNI-Deadlocks und Abbrüche im Android-Binder-Thread-Pool (*"connect() failed / No card present"*).
3. **Sonderzeichen-Mutation beim NDEF-Parsing:** Der standardkonforme NDEF-Schreibcode brennt die Daten inklusive Record-Headern und dem NFC-Forum-Terminator-Byte (0xFE) auf den NTAG215-Chip. Da die ursprüngliche Lese-Schleife den Sektor stur als rohen UTF-8-String auslas, interpretierte sie die Steuerbytes am Anfang als Zeichensalat (&´┐¢"Tde) und das 0xFE am Ende als ungültiges Zeichen. Dies führte beim anschließenden Parsen in Dart zu einer FormatException.
4. [TOC]

### 2.2 Der steinige Weg zur Lösung: Was alles versucht wurde

- **Versuch 1 (Software-Resets via Escape-APDUs):** Es wurde versucht, den Reader über proprietäre ACS-Escape-Kommandos (Klasse E0) im laufenden Betrieb zu resetten oder den Standby-Timer per Software zu überschreiben. Diese Befehle wurden jedoch im mobilen Betrieb blockiert.
- **Versuch 2 (Kondensatoren-Entladung ohne Öffnen):** Ein 60-sekündiges Halten aller Tasten ohne Kabel wurde getestet, um die Kondensatoren zu leeren. Aufgrund der permanenten Spannungsversorgung durch den internen Akku blieb das RAM jedoch aktiv und der Freeze bestehen.
- **Versuch 3 (Umschalten via Windows-HID-Tool):** Das herstellereigene *Keyboard Configuration Tool* unter Windows wurde verwendet, um den Modus auf *CCID* umzustellen. Das Tool schickte jedoch nur Buzzer-Abfragen an den Chip und veränderte das primäre Boot-Register nicht dauerhaft.

### 2.3 Die endgültige Implementierung im Code

Die finale, stabile Lösung wurde durch eine dreistufige Synchronisation direkt in unserem nativen Quellcode (Kotlin/Dart) realisiert:

**Schritt A: Modus-Zwang im nativen Boot-Block (Kotlin)**

Sobald die App den Reader über den DIRECT-Kanal initiiert, beschreiben wir das herstellerspezifische Bluetooth-Betriebsmodus-Register 40h im NVRAM des Controllers und zwingen das Gerät permanent in den reinen **CCID-Smartcard-Reader-Modus** (01h). Dies tilgt die HID-Abschaltfehler restlos:

```
val directCard = activeTerminal!!.connect("DIRECT")

// Register 40h (Bluetooth Mode) -> 01h (Reiner CCID
Reader Modus)

val cmdForceCcidMode = byteArrayOf(0xE0.toByte(), 0x00, 0x00,
0x40.toByte(), 0x01, 0x01.toByte())

directCard.transmitControlCommand(3500, cmdForceCcidMode)
```

**Schritt B: Die native Polling-Bremse via MethodChannel**

Um die GATT-Kollisionen beim Schreiben zu verhindern, haben wir eine native Sperre (setWritingMode) via MainActivity.kt etabliert. Vor dem Start der Schreibschleife friert Dart das reaktive Hintergrund-Polling in Kotlin starr ein, sodass der Schreibkanal exklusiven und ungestörten Zugriff auf die Karte erhält:

```
// Vor dem Schreibvorgang im NfcService (Dart):

await _acsChannel.invokeMethod('setWritingMode', {'enabled':
true});

// ... atomarer Schreibbefehl 'writeTag' über den exklusiven
Kanal ...

await _acsChannel.invokeMethod('setWritingMode', {'enabled':
false});
```

**Schritt C: NDEF-tolerantes Parsing (Dart)**

Die Auswertungslogik in Dart wurde so umgebaut, dass sie unabhängig von den ausgelesenen Sektor-Grenzen dynamisch nach unserem Protokoll-Präfix DP1; sucht. Vorgelagerte NDEF-Header-Bytes werden abgeschnitten, und nachfolgender Zeichensalat (wie das 0xFE-Endbyte) wird über einen regulären Ausdruck (RegEx) restlos eliminiert, bevor die Namensfelder in den Speicher wandern.

**TEIL 3: Support & Bezugsquellen für den Patch**

Sende für die Anforderung der fehlerfreien Firmware eine E-Mail auf Englisch an den offiziellen ACS-Support. Bitte achte unbedingt darauf, die Platzhalter am Ende des Textes durch deine tatsächliche Modellbezeichnung sowie deine aktuell installierte Firmware-Version zu ersetzen. Nur so kann der Support prüfen, ob die Datei exakt zu deiner Hardware-Revision passt, und dir das richtige Update-Paket zusenden.

**E-Mail-Text (Kopiervorlage):**

**Empfänger:** support@acs.com.hk

**Betreff:** Firmware Update Request for ACR1555U-A1 (Fix for BLE-HID Shutdown Freeze)

Dear ACS Support Team,

I am using the mobile smart card reader ACS ACR1555U-A1 for a fire service operation application (Revision: RR629-002832).

We are currently experiencing a severe hardware lockup/freeze issue: Whenever the device disconnects from Bluetooth or reaches its standby timeout, it shuts down with a double-beep. After this, the power button becomes completely unresponsive. The device can only be revived by physically disconnecting and reconnecting the internal battery cable from the PCB.

According to technical reference, this issue regarding the BLE-HID and PMIC register conflict is fully resolved in Firmware Version 1.03.02.

Could you please provide us with the official Firmware Update Tool and the firmware files for Version 1.03.02 (or newer) to flash our device via Windows?

Device Information:

- Model: [Hier genaue Modellbezeichnung eintragen, z.B. ACS ACR1555U-A1]
- Current Firmware: [Hier deine aktuelle Version eintragen, z.B. 1.02.08]

Thank you very much for your support.

Best regards,

[Dein Name / Deine Feuerwehr]