Mobile Unique ID via NFC
- Source code software examples -
v1.0

Table of contents

Table of contents 2

About 3

Usage 3

AID usage 5

Revision history 8

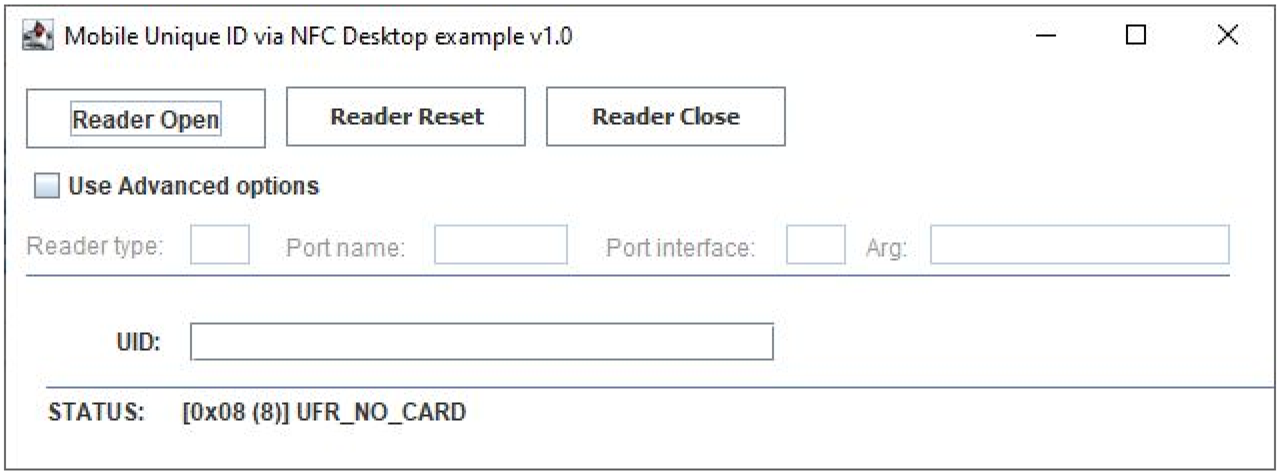
About

This document demonstrates the use of "Mobile Unique ID via NFC" software written in Java for desktop and Android. The purpose of these software examples is to obtain the Android device's Unique Identifier and read it using the μFR Series NFC readers.

Java desktop example:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/ufr-mobile_unique_id_via_nfc-examples-java.git
Java Android example:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/ufr-mobile_unique_id_via_nfc-examples-android

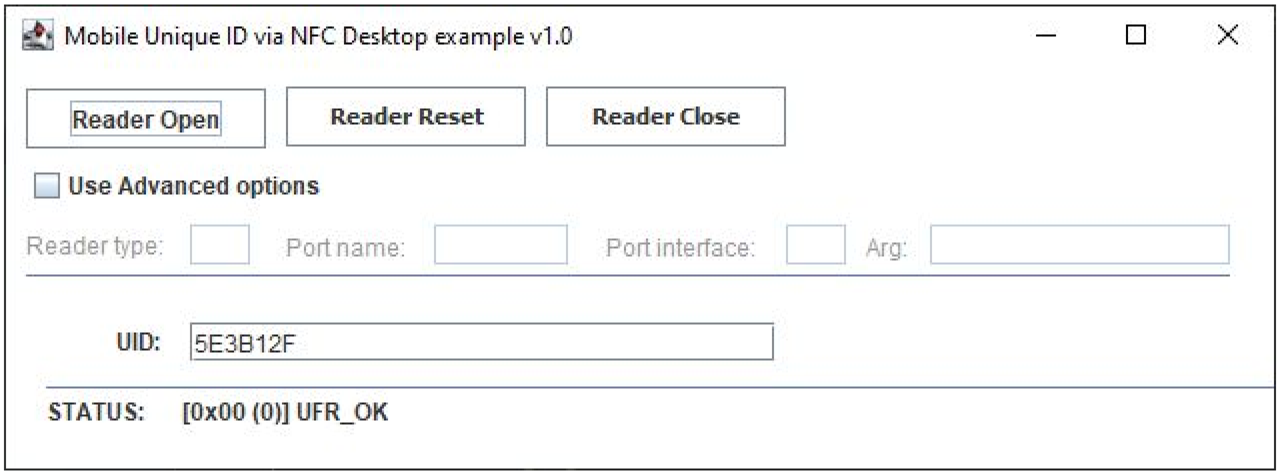
Usage

Connect the μFR Series NFC reader to your PC and run the desktop software example.
Click the "Reader Open'' button to establish communication with the μFR Series NFC reader.
Successful "Reader Open" will start a command transmission loop for obtaining IDs from NFC tags and Android devices. Clicking the "Reader Close" button or closing the application windows will end the loop.



STATUS: [0x08 (8)] UFR_NO_CARD

When no card is present.



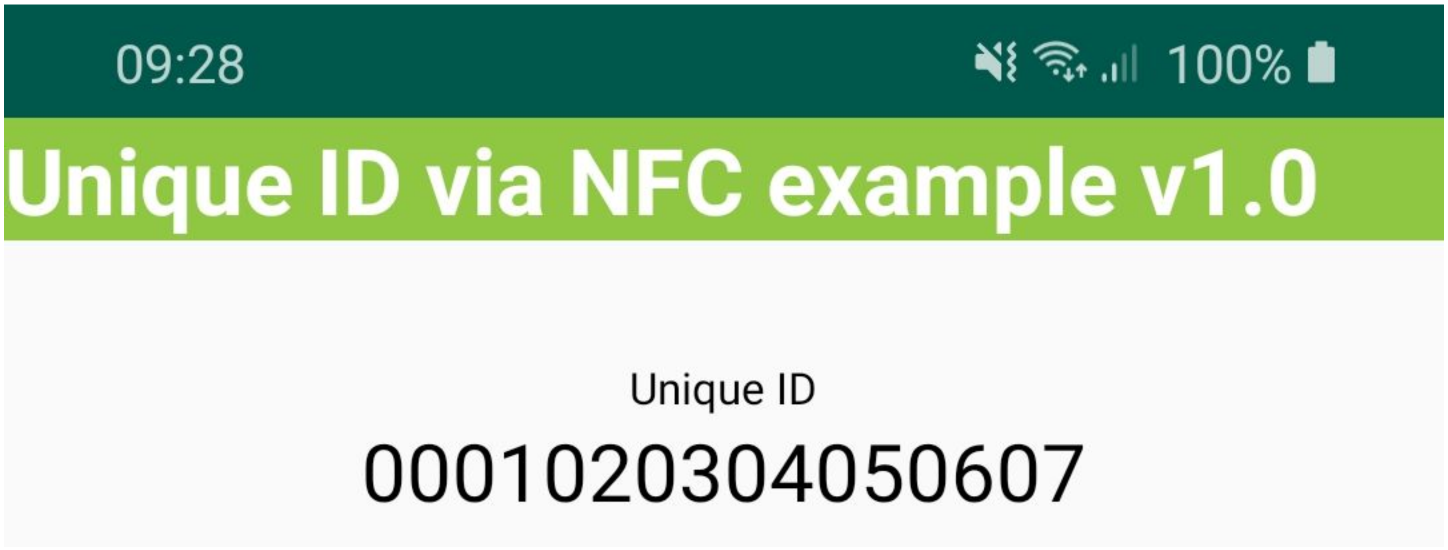
UID: 5E3B12F

STATUS: [0x00 (0)] UFR_OK

Tag with a 4-byte UID is detected.

To receive a Unique identifier from an Android device, you will need to install our Android app first. "Mobile Unique ID via NFC" Android app is available for download at:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/ufr-mobile_unique_id_via_nfc-examples-android

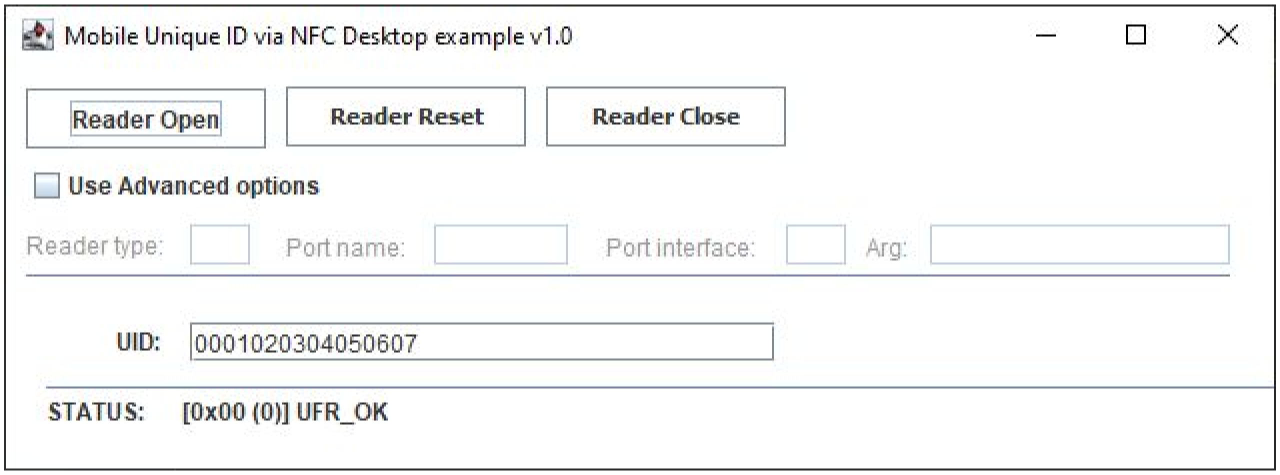
This app will register an HCE (Host Card Emulation) based service that responds to a particular AID sent from our desktop software. As a result, the response will consist of an 8-byte UID of the Android device.



Unique ID

0001020304050607

Android UID is displayed when Android app example is launched



UID: 0001020304050607

STATUS: [0x00 (0)] UFR_OK

Successfully read Android UID displayed in the desktop software

“Mobile Unique ID via NFC” Android app registers a service on the Android device with a specific AID. Therefore, this app needs to be installed only once. open/run the app alongside the desktop software afterward. Also, the service used in this software example allows you to read the Android UID even when the device is locked. So, there is no need to unlock the device first. Simply tap your Android phone on the µFR Series NFC reader.

For additional references please check the official Android API:
https://developer.android.com/reference/android/provider/Settings.Secure#ANDROID_ID

Values of received UID are scoped by signing key and user. The value may change if a factory reset is performed on the device or if an APK signing key changes. For more information about how the platform handles this type of UID in Android 8.0 (API level 26) and higher, see Android 8.0 Behavior Changes.

AID usage

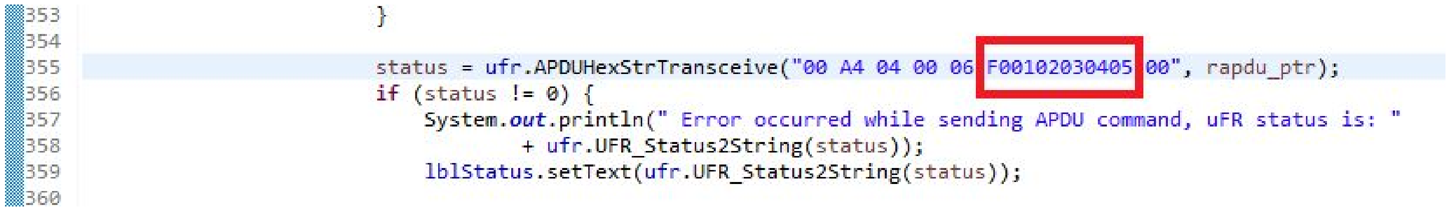
The rules for smartcard application identifiers (AIDs) are defined in ISO/IEC 7816-4. An AID has at least 5 bytes and may consist of up to 16 bytes. Based on the first 4 bits, AIDs are divided into different groups. The most relevant groups defined in ISO/IEC 7816-4 are:

AIDs starting with 'A': internationally registered AIDs
AIDs starting with 'D': nationally registered AIDs
AIDs starting with 'F': proprietary AIDs (no registration)

For (inter)nationally registered AIDs, the AID is split into two parts: a 5-byte mandatory RID (registered application provider identifier), and an optional PIX (proprietary application identifier extension) of up to 11 bytes.

For proprietary AIDs (F...), you can use any arbitrary value.
AID used in this example is the same one used in our previous Android HCE example:
https://www.d-logic.net/code/nfc-rfid-reader-sdk/ufr-examples-android-host_card_emulation.git

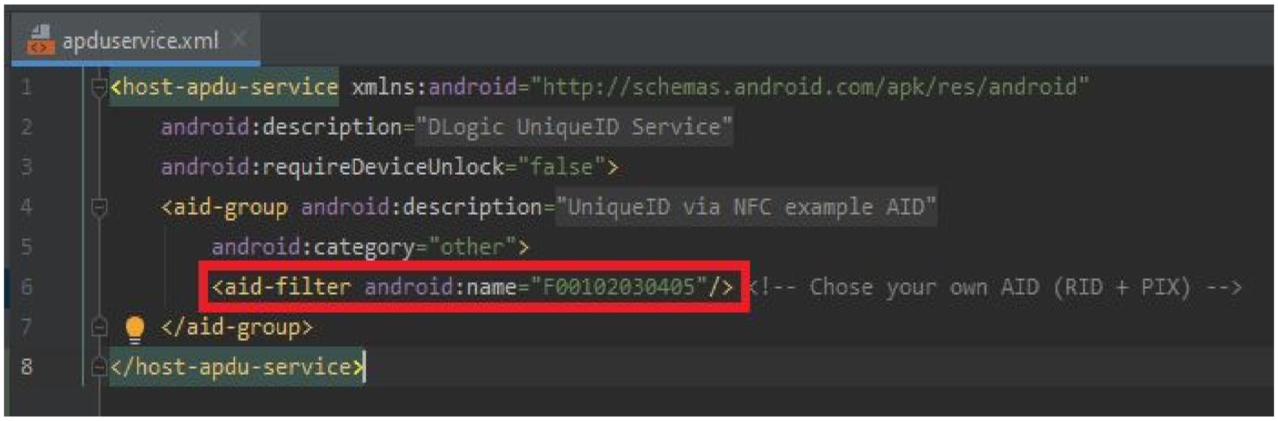
- To change this AID, following changes are mandatory:
  In desktop software, change the AID that is being sent with the **APDUHexStrTransceive()** function. This change should be introduced on the Line 355 in the "**window.java**" file (full path: ufr_mobile_unique_id_via_nfc_examples_java/src/**window.java**)



```
353                    }
354
355                    status = ufr.APDUHexStrTransceive("00 A4 04 00 06 F00102030405 00", rapdu_ptr);
356                    if (status != 0) {
357                        System.out.println(" Error occurred while sending APDU command, uFR status is: "
358                                + ufr.UFR_Status2String(status));
359                        lblStatus.setText(ufr.UFR_Status2String(status));
360
```

Current AID being sent from the example, code snippet.

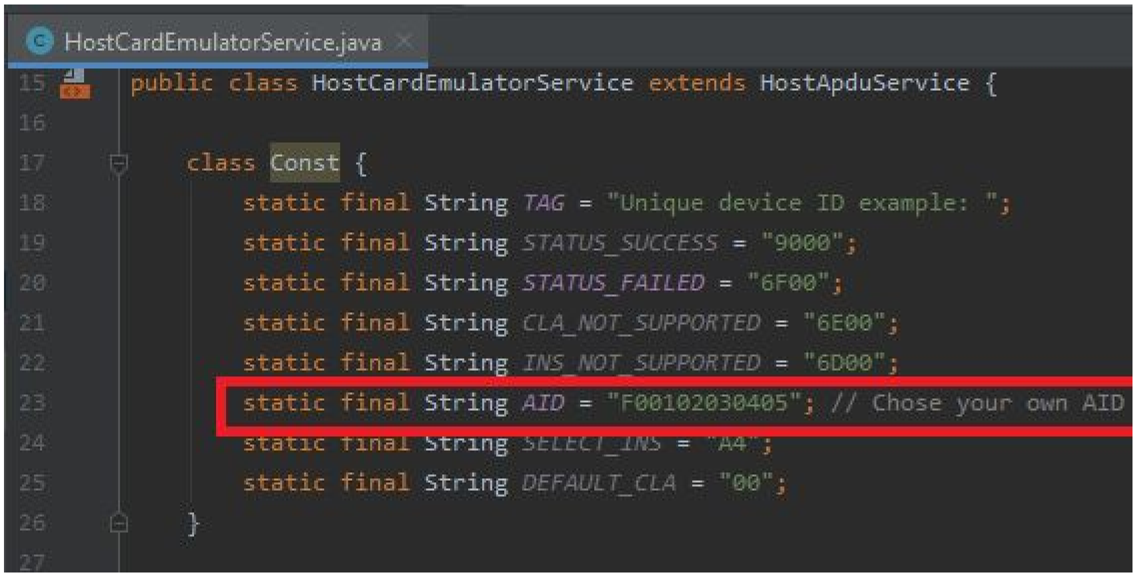
- Also, AID must be changed in the Android app example. So that the AID sent from desktop software corresponds to the one registered in the Android app. Introduce the following changes in the Android example:



apduservice.xml

```
<host-apdu-service xmlns:android="http://schemas.android.com/apk/res/android"
    android:description="DLogic UniqueID Service"
    android:requireDeviceUnlock="false">
    <aid-group android:description="UniqueID via NFC example AID"
        android:category="other">
        <aid-filter android:name="F00102030405"/> <!-- Chose your own AID (RID + PIX) -->
    </aid-group>
</host-apdu-service>
```

Change the registered AID in the "**apudservice".xml** file.



HostCardEmulatorService.java

```
public class HostCardEmulatorService extends HostApduService {

    class Const {
        static final String TAG = "Unique device ID example: ";
        static final String STATUS_SUCCESS = "9000";
        static final String STATUS_FAILED = "6F00";
        static final String CLA_NOT_SUPPORTED = "6E00";
        static final String INS_NOT_SUPPORTED = "6D00";
        static final String AID = "F00102030405"; // Chose your own AID
        static final String SELECT_INS = "A4";
        static final String DEFAULT_CLA = "00";
    }

```

Change the “**AID”** constant value in the “**HostCardEmulatorService.java”** file**.**

Revision history

Date	Version	Comment
2021-02-08	1.0	Base document