

ABHA Face Authentication

Android Implementation Guide

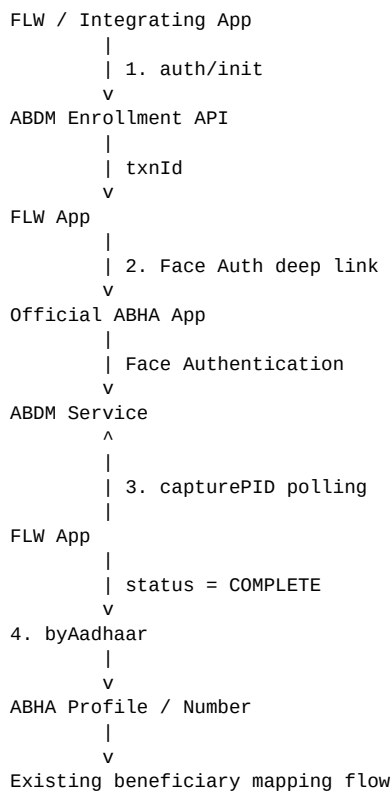
Reusable developer documentation for FLW and other applications with the same architecture, UI and ABHA flow.

1. Overview

This guide describes the Android implementation for creating an ABHA using Face Authentication. The integrating app initializes an ABDM transaction, opens the official ABHA application through a Face Auth deep link, polls the transaction state, and calls the final enrollment API after successful authentication.

Face capture/authentication is performed by the official ABHA application. The integrating app does not directly capture the Face Auth PID.

2. End-to-End Flow



3. Responsibility Split

Component	Responsibility
Integrating App	Initialize transaction, launch ABHA app, poll status, call enrollment API, handle errors and map ABHA to the application's b
Official ABHA App	Perform face capture/authentication and submit authentication information to ABDM.
ABDM APIs	Maintain transaction state and process final ABHA enrollment.

4. Step 1 — Initialize Face Auth

Sandbox: POST <https://abhasbx.abdm.gov.in/abha/api/v3/enrollment/enrol/auth/init>

```
{
  "scope": [
    "abha-enrol",
```

```
        "face-auth"
    ]
}
```

Example response:

```
{
  "txnId": "96d21b5c-22b3-484e-b936-06f3735b57d4",
  "message": "Transaction Id generated Successfully"
}
```

Always use the txnId returned by auth/init. Do not generate a local/random transaction ID.

5. Step 2 — Launch Official ABHA App

Environment	Face Auth URL	Package ID
Sandbox	https://phrsbx.abdm.gov.in/face-auth?txnId=<txnId>	in.ndhm.phr.debug
Production	https://phr.abdm.gov.in/face-auth?txnId=<txnId>	in.ndhm.phr

Sandbox Kotlin:

```
val intent = Intent(
    Intent.ACTION_VIEW,
    Uri.parse("https://phrsbx.abdm.gov.in/face-auth?txnId=$txnId")
).apply {
    setPackage("in.ndhm.phr.debug")
}
context.startActivity(intent)
```

Production Kotlin:

```
val intent = Intent(
    Intent.ACTION_VIEW,
    Uri.parse("https://phr.abdm.gov.in/face-auth?txnId=$txnId")
).apply {
    setPackage("in.ndhm.phr")
}
context.startActivity(intent)
```

Use ACTION_VIEW with the documented deep link. Do not depend on an internal ABHA Activity name.

6. ABHA App Knowledge Verified During FLW Implementation

Environment	Package	Face Auth handler
Sandbox	in.ndhm.phr.debug	phrsbx.abdm.gov.in/face-auth
Production	in.ndhm.phr	phr.abdm.gov.in/face-auth

Both installed apps were verified to expose the Face Auth URL through an ACTION_VIEW HTTP/HTTPS intent filter. The Production package is in.ndhm.phr; the Sandbox package is in.ndhm.phr.debug.

7. Checking ABHA App Availability

If an explicit installation/package check is required, declare both packages under using Android package visibility queries:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">

    <queries>
        <package android:name="in.ndhm.phr.debug" />
        <package android:name="in.ndhm.phr" />
    </queries>

    <application
        ...>
    </application>

</manifest>
```

For this feature, resolving the exact Face Auth intent is preferable because it verifies the capability actually required by the flow:

```
fun canOpenAbhaFaceAuth(
    context: Context,
```

```

        packageName: String,
        faceAuthUrl: String,
        txnId: String
    ): Boolean {
        val intent = Intent(
            Intent.ACTION_VIEW,
            Uri.parse("$faceAuthUrl?txnId=$txnId")
        ).apply {
            setPackage(packageName)
        }
        return intent.resolveActivity(context.packageManager) != null
    }
}

```

8. Step 3 — User Performs Face Authentication

```

FLW App
|
v
ABHA Face Auth Screen
|
v
User completes Face Authentication
|
v
ABHA App submits authentication information
|
v
ABDM transaction state updated

```

Opening the ABHA app is not success. The transaction status returned by capturePID is the source of truth.

9. Step 4 — Poll capturePID

Sandbox: POST <https://abhasbx.abdm.gov.in/abha/api/v3/enrollment/enrol/capturePID>

```

{
  "scope": [
    "abha-enrol",
    "face-verify"
  ],
  "txnId": "<txnId>"
}

```

Poll approximately every 5–10 seconds.

Status	Action
PENDING	Wait and poll again.
VERIFIED	Wait and poll again.
COMPLETE	Stop polling and call byAadhaar.
FAILED	Stop polling and handle failure/retry.

Only COMPLETE should trigger the final enrollment request.

10. Step 5 — byAadhaar

Sandbox: POST <https://abhasbx.abdm.gov.in/abha/api/v3/enrollment/enrol/byAadhaar>

```

{
  "authData": {
    "authMethods": [
      "face_auth"
    ],
    "face": {
      "txnId": "<txnId>",
      "aadhaar": "<encrypted_aadhaar>",
      "mobile": "<mobile_number>"
    }
  },
  "consent": {
    "code": "abha-enrollment",
    "version": "1.4"
  }
}

```

Field	Requirement
authMethods	Must contain face_auth.
face.txnId	Same txnId returned by auth/init.
face.aadhaar	Encrypted Aadhaar number.
face.mobile	Mobile number.
consent.code	abha-enrollment.
consent.version	1.4.

Important: Aadhaar is encrypted. Mobile is passed as the mobile number; it is not an encrypted Aadhaar field.

11. Successful Enrollment

The response provides ABHA profile information, including the ABHA number/address and account status. The exact response can vary depending on whether an ABHA already exists.

```
{
  "ABHAProfile": {
    "preferredAddress": "xxxx@sbx",
    "firstName": "Kushal",
    "lastName": "Pandita",
    "dob": "11-02-2000",
    "gender": "M",
    "mobile": "xxxxxxxxxx",
    "mobileVerified": true,
    "ABHANumber": "91-****-****-8769",
    "abhaStatus": "ACTIVE"
  },
  "isNew": false
}
```

Pass the resulting ABHA information into the application's existing beneficiary/user mapping flow.

12. Complete State Machine

```
auth/init
|
v
txnId
|
v
Launch ABHA Face Auth
|
v
capturePID polling
|
+--> PENDING ----+
|                   |
+--> VERIFIED ----+--> wait 5-10 sec --> poll again
|
+--> COMPLETE --> byAadhaar --> ABHA result
|
+--> FAILED ----> stop --> retry with a new transaction
```

13. Transaction Rules

Treat one txnId as one Face Authentication attempt. If the attempt fails or times out, discard the attempt and start a new auth/init transaction.

Do not assume that returning from the ABHA app means success. A user may cancel or fail Face Auth. The application should rely on capturePID status.

14. Sandbox vs Production

Item	Sandbox	Production
Face Auth URL	https://phrsbx.abdm.gov.in/face-auth	https://phr.abdm.gov.in/face-auth

Item	Sandbox	Production
ABHA package	in.ndhm.phr.debug	in.ndhm.phr
ABHA app	ABHA Sandbox	Official ABHA app
Enrollment host	abhasbx.abdm.gov.in	Configured production ABDM enrollment host

Keep environment-specific values in build/configuration settings instead of hard-coding them across the feature.

15. Suggested Android Architecture

```

UI
|
v
ViewModel / UseCase
|
+-- initiateFaceAuth()
+-- launchFaceAuth()
+-- pollCapturePid()
+-- enrollUsingFaceAuth()
|
v
Repository
|
v
ABDM API Service

```

Because the other applications already have the same architecture and existing ABHA flow, this feature should be ported as a reusable feature. The application-specific part is primarily the final beneficiary/user mapping and UI entry point.

16. Error Handling

Scenario	Recommended handling
ABHA app unavailable	Stop before starting Face Auth and inform the user.
User exits ABHA	Do not assume success; resume/check transaction state.
Face Auth FAILED	Stop polling and allow retry with a new transaction.
Transaction timeout	Stop polling and create a new transaction for the next attempt.
byAadhaar failure	Handle ABDM error; do not mark ABHA creation successful.

17. Important Implementation Learnings

- QR is not required for same-device Android integration; the Intent/deep-link approach is suitable.
- Do not directly launch an ABHA internal Activity; use ACTION_VIEW with the Face Auth URL and correct package.
- Never generate txnId locally.
- ABHA app opened ≠ Face Auth success.
- Proceed to byAadhaar only when capturePID is COMPLETE.
- Sandbox and Production have different package IDs.
- Package visibility can affect PackageManager installation checks on modern Android.

18. Developer Checklist

- auth/init integrated with scope [abha-enrol, face-auth]
- txnId stored for the current attempt
- Correct Sandbox/Production Face Auth URL configured
- Correct ABHA package configured
- ABHA availability handled
- ACTION_VIEW deep link implemented
- capturePID polling implemented at 5–10 second interval

- PENDING, VERIFIED, COMPLETE and FAILED handled
- Polling stops on terminal state
- byAadhaar called only after COMPLETE
- authMethods = [face_auth]
- Aadhaar encrypted correctly
- Mobile passed as mobile number
- consent.code = abha-enrollment
- consent.version = 1.4
- ABHA response handled
- Existing beneficiary/user mapping integrated
- Retry/new transaction handling implemented
- Sandbox tested
- Production ABHA app tested

19. Official ABDM References

Primary: **ABHA Creation via Face Auth (QR based)**

https://sandboxcms.abdm.gov.in/uploads/Face_Auth_ABHA_Creation_Steps_30_07_2025_6282bda40f.pdf

Related: **Find ABHA using Face / QR Scan API Specification**

https://sandboxcms.abdm.gov.in/uploads/Updated_Find_ABHA_Face_QR_Scan_API_Specification_33413c606e.pdf

Official: **ABDM FAQ**

<https://abdm.gov.in/FAQ>

The official ABDM documents are the authoritative references for API contracts and protocol behavior. Package IDs and Android deep-link behavior documented here were additionally verified during the FLW implementation.

20. Porting to Other Applications

1. Add environment configuration
2. Add/verify ABDM API interfaces
3. Add Face Auth repository/use-case
4. Add capturePID polling
5. Add ABHA deep-link launcher
6. Add ViewModel/UI state handling
7. Connect successful response to existing ABHA mapping
8. Add UI entry point
9. Test Sandbox
10. Verify Production ABHA app flow

The other applications should therefore require a feature-port and integration with their existing beneficiary flow, not a completely new ABHA integration design.