SDK and Mobile Integration Reference

3.5.1 Overview

The solution provides a suite of native Software Development Kits (SDKs) that enable mobile banking applications to securely integrate with the CIAM and Digital Trust platform. These SDKs encapsulate complex security protocols, device attestation logic, and authentication user interfaces into cohesive, well-defined components.

The primary SDKs available are:

SDK	Primary Function	Integration Model
HAAPI SDK	API-driven authentication orchestration, Passkey/WebAuthn support, and CIBA-based push challenge handling.	Embedded library; exposes hypermedia client APIs.
Mobile SDK	Turnkey UI flows for service registration, transaction authorization, PIN management, and biometric settings.	Embedded library with pre-built UI; invoked via encrypted Intent/Delegate.
Device Intelligence SDK	Collection of device telemetry, integrity signals, and fingerprint hashes for risk evaluation and trusted device binding.	Embedded module (often bundled with HAAPI or Mobile SDKs); transparent background operation.

3.5.2 HAAPI SDK

The Hypermedia Authentication API (HAAPI) SDK is the native client implementation of Curity's API-driven authentication model. It enables mobile applications to execute OAuth 2.0 and OpenID Connect flows without relying on embedded web views, providing a fully native and secure user experience.

Core Capabilities:

Capability	Description
Hypermedia-Driven Flow	The SDK interacts with the Curity server using a stateful, hypermedia-based API where each step in the authentication journey is described by server responses.
Passkey / WebAuthn	Native integration with platform biometrics (Face ID, Touch ID, Android Biometric) for creating and asserting Passkeys.
Push Challenge Handling	Enables the application to receive, display, and respond to CIBA-based push challenges for transaction approval and step-up authentication.
OAuth 2.0 / OIDC Compliance	Implements standard flows including Authorization Code with PKCE, Client Credentials, and Device Flow.

Supported Platforms: Android, iOS, Web (JavaScript)

3.5.3 Mobile Banking SDK

The **Mobile Banking SDK** is a specialized, turnkey component that provides a complete, branded user interface for all related operations. It abstracts the underlying business logic and secure communication with the backend gateway.

Functional Modules:

Module	Description
User Enrollment & Registration	Guides the user through initial onboarding, including identity verification (e.g., eKYC), account activation, and enrollment of authentication methods such as password, passkey, or biometric authentication.
Authentication & Login	Handles user login flows, including primary authentication (e.g., password, passkey) and step-up authentication (e.g., biometric approval, push confirmation, or multi-factor authentication) based on risk and policy.

Module	Description
Authentication Factor Lifecycle Management	Manages the lifecycle of authentication factors, including creation, update, reset, recovery (e.g., forgotten credentials), and expiration policies for passwords, PINs, passkeys, or other factors.
Device & Session Management	Provides capabilities to manage trusted devices and active sessions, including viewing registered devices, revoking sessions, enabling/disabling device trust, and controlling device-based access.
Biometric & Passkey Configuration	Allows users to configure and manage device-native authentication methods such as biometrics (fingerprint, face recognition) and passkeys (FIDO2/WebAuthn) as secure alternatives to traditional credentials.

Integration Pattern: The SDK is distributed as a compiled binary (.aar for Android, .framework for iOS) and is invoked by the host banking application. All communication between the host application and the SDK is encrypted using a pre-shared AES-256-GCM key.

3.5.4 Device Intelligence / Fingerprint Module

The **Device Intelligence Module** is responsible for the secure collection of device-specific telemetry and integrity signals used to build a trusted device profile. It is typically embedded within the HAAPI or SDKs and operates transparently during SDK initialization or authentication flows.

Data Categories Collected:

Category	Examples
Device Identifiers	DEVICE_ID (generated and stored in Keystore/Keychain), BRAND_NAME, MODEL
Operating System & App Context	OS, OS_VERSION, APP_VERSION, SECURITY_PATCH

Category	Examples
Integrity Signals	IS_ROOTED, IS_EMULATOR, IS_HOOK_DETECTED, IS_APP_TAMPERED
Network Context	IP_ADDRESS, VPN_DETECTED, PROXY_DETECTED, TOR_DETECTED
Fingerprint Hashes	COMPOSITE_HASH, HW_HASH, SW_HASH

The collected data is transmitted to the backend as part of the device context in API requests and is ultimately stored in the DEVICES table (see Section 3.1) for policy evaluation and risk scoring.

3.5.5 Communication Security

All inter-process communication between the host application and the Mobile SDK is encrypted using **AES-256 in Galois/Counter Mode (GCM)** .

Security Aspect	Implementation
Encryption Algorithm	AES-256-GCM with a 128-bit initialization vector (IV).
Key Management	A pre-shared symmetric key and IV are injected into the host application at build time via secure environment variables or properties files. These values are never hardcoded.
Data Flow	The host application serializes the request to JSON, encrypts the payload, and passes the ciphertext to the SDK. The SDK decrypts the request internally, processes it, and returns an encrypted response.

3.5.6 Environment and Platform Requirements

The following table outlines the minimum supported versions and development environment requirements for integrating the SDKs.

Requirement	**Android**	**iOS**
Minimum OS Version	Android 8.0 (API Level 26)	iOS 14.0
Development Environment	Android Studio Giraffe (2022.3.1)+	Xcode 16.0+
Primary Language	Kotlin 2.0+	Swift 5.0+
Build System	Gradle 8.0+	Swift Package Manager / CocoaPods
Network	HTTPS connectivity to backend services is required.	

3.5.7 Security Considerations

When integrating the provided SDKs, the following security practices must be observed to maintain the overall security posture of the banking application.

Consideration	**Requirement**
Secrets Management	AES_KEY, AES_VECTOR, consumerKey, and consumerSecret must be injected at build time from secure environment variables or a secrets vault. Hardcoding these values is prohibited.

Consideration	Requirement
Certificate Pinning	It is strongly recommended to implement certificate pinning for all HTTPS connections to the backend services to mitigate Man-in-the-Middle (MITM) attacks.
ProGuard / R8 (Android)	ProGuard rules must be configured to preserve all SDK classes and methods. Failure to do so will cause runtime exceptions.
Root/Jailbreak Detection	The host application should respect the risk assessment provided by the Device Intelligence Module and consider restricting functionality on compromised devices as per bank policy.