APIs for Device Management

These APIs support the registration, querying, updating, and revocation of devices, combining standard CIAM interfaces with cryptographically secured flows for high-assurance device binding.

4.2.1 API Summary

API	Provider	Method & Endpoint	Purpose	Security
List Devices	Curity	GET /Devices?filter=userName eq "{user}"	Retrieves a paginated list of authentication devices associated with a user.	Bearer JWT
Get Device	Curity	GET /Devices/{id}	Retrieves detailed information of a specific authentication device, including status and metadata.	Bearer JWT
Register Device	Curity	POST /Devices	Registers a new authentication device using the SCIM Device schema (initial state: PENDING).	Bearer JWT
Update Device	Curity	PATCH /Devices/{id}	Updates mutable attributes of a device (e.g., display name, status, trust level).	Bearer JWT
Delete Device	Curity	DELETE /Devices/{id}	Permanently removes (DEREGISTERED) a device from the user profile.	Bearer JWT
GraphQL Device Query	Curity	POST /graphql	Provides flexible queries to retrieve device information with nested relationships (e.g., sessions, authentication factors).	Bearer JWT

API	Provider	Method & Endpoint	Purpose	Security
Check Device Context	Auth Service	POST /auth/device/check-context	Evaluates device context (recognized, trusted, risk signals) during login/authentication flow.	JWS + JWE
Register Device (Secure)	Auth Service	POST /auth/device/register	Registers and binds a device with cryptographic keys (device binding / passkey / secure key pair).	JWS + JWE
Confirm Device Enrollment	Auth Service	POST /auth/device/confirm	Confirms device enrollment (e.g., user approval or step-up authentication), transitioning device from PENDING → ACTIVE.	JWS + JWE
List Devices (Auth Service)	Auth Service	GET /auth/device/list	Returns a simplified list of devices with status (ACTIVE, INACTIVE, LOCKED, etc.) for user self-service.	Bearer JWT
Update Device Status	Auth Service	POST /auth/device/update-status	Updates device state (e.g., ACTIVE → INACTIVE, LOCKED, or reactivation after verification).	JWS + JWE
Deregister Device	Auth Service	POST /auth/device/deregister	Permanently removes a device (status → DEREGISTERED) and revokes associated sessions.	JWS + JWE

4.2.2 Detailed API Descriptions

- **Standard CIAM Device APIs (Curity)**

The solution exposes a set of standard **SCIM 2.0** /Devices endpoints, providing a RESTful interface for device lifecycle management. These APIs are ideal for integration with

administrative portals, self-service account management consoles, and provisioning systems.

- GET /Devices: Supports filtering by attributes such as userName and displayName. The response is a ListResponse containing an array of Device resources, each with fields like deviceId, displayName, status, lastActivity, and os.
- POST /Devices: Creates a new device record. The request body conforms to the SCIM Device schema. Upon success, the device is associated with the specified user and assigned a unique id.
- PATCH /Devices/{id}: Allows partial updates to a device. Common operations include updating the friendly name (displayName) or changing the status to REVOKED to disable the device.
- DELETE /Devices/{id}: Permanently removes the device record from the registry.
- **GraphQL Device Query (Curity)**

For flexible data retrieval, a **GraphQL** endpoint is available. The following query retrieves a user's devices along with their status, last activity, and operating system details.

Secure Device Registration APIs (RAS Service)

For high-assurance device binding, the solution provides a dedicated set of APIs via the **RAS Service**. These APIs incorporate **JWS (JSON Web Signature)** for integrity and **JWE (JSON Web Encryption)** for confidentiality, ensuring that sensitive key material is protected during device onboarding.

Check Device Status

- **Endpoint:** POST /ras/device/check-status
- **Purpose:** Determines if a device is eligible for registration. The response indicates the current device state (UNREGISTERED, ACTIVE, LOCKED) and whether registration is permitted.

Register Device

- **Endpoint:** POST /ras/device/register
- **Purpose:** Submits the device's public keys (for encryption and signature verification) along with hardware and software metadata. The device is placed in a PENDING state awaiting OTP verification.

- **Request Payload (Encrypted):** Includes userId, deviceId, deviceName, and an array of deviceKeys (each specifying keyAlgo, keyLength, publicKey, profileId, keyUse).

Validate Device OTP

- **Endpoint:** POST /ras/device/validate-otp
- **Purpose:** Completes the registration by validating the OTP sent to the user. Upon success, the device transitions to ACTIVE and the user's secret key for TOTP/OCRA generation is returned.

List, Update, and Unregister Devices

- GET /ras/device/list: Returns a simplified list of devices and their statuses for the authenticated user.
- POST /ras/device/update-status: Updates the status of a registered device (e.g., from ACTIVE to INACTIVE).
- POST /ras/device/unregister: Permanently revokes a device, removing its ability to authenticate or authorize transactions.

4.2.3 Flow Integration

A typical high-assurance device registration flow using the RAS Service APIs proceeds as follows:

1. The mobile SDK calls /ras/key/server-sessionkey to establish a secure session.
2. The SDK calls /ras/device/check-status to verify eligibility.
3. If eligible, the SDK calls /ras/device/register to submit device keys.
4. The user receives and enters an OTP; the SDK calls /ras/device/validate-otp to complete activation.

APIs for Authentication Flow

These APIs span the OAuth 2.0 / OpenID Connect layer, the Risk Engine's policy decision point, and the transaction orchestration layer.

4.3.1 API Summary

API	Provider	Method & Endpoint	Purpose	Security
OAuth2 Token	Curity	POST /oauth2/token	Issues access tokens using various grant types.	Client Auth
OAuth2 Revoke	Curity	POST /oauth2/revoke	Revokes an access or refresh token.	Client Auth
OAuth2 Introspect	Curity	POST /oauth2/introspect	Validates the state and scope of an access token.	Client Auth
CIBA Backchannel Auth	Curity	POST /bc-authorize	Initiates a decoupled authentication flow (e.g., Push Approval).	Client Auth
Device Flow (QR Login)	Curity	POST /device	Initiates the OAuth 2.0 Device Authorization Grant for QR login.	Public Client

API	Provider	Method & Endpoint	Purpose	Security
Risk Evaluate	Risk Engine	POST /api/v1/risk/evaluate	Evaluates risk score and context to determine required authentication factors.	X-Trace-Id
Risk Fulfill	Risk Engine	POST /api/v1/risk/decision/{id}/fulfill	Confirms that step-up authentication has been completed.	X-Trace-Id
Get Decision Status	Risk Engine	GET /api/v1/risk/decision/{id}	Queries the current state of a risk decision.	X-Trace-Id
Get Server Session Key	RAS Service	GET /ras/key/server-sessionkey	Retrieves server public keys and a session identifier for securing subsequent calls.	Bearer JWT

API	Provider	Method & Endpoint	Purpose	Security
Create Auth Request	RAS Service	POST /ras/auth-request	Initiates a transaction authorization request from the bank's backend.	mTLS / TLS
Get Auth Request Details	RAS Service	GET /ras/auth-request/{sessionId}	Retrieves transaction details for user confirmation.	Bearer JWT
Validate Transaction OTP	RAS Service	POST /ras/transaction/validate-otp	Validates the TOTP/OCRA code entered by the user.	JWS + JWE
Validate Basic OTP	RAS Service	POST /ras/validate-basic-otp	Validates a basic OTP (SMS/Email) for step-up or registration.	Bearer JWT

4.3.2 Detailed API Descriptions

- **OAuth 2.0 / OpenID Connect APIs (Curity)**

The solution includes a fully compliant OAuth 2.0 Authorization Server and OpenID Connect Provider. The following endpoints form the foundation for secure client authentication and token management.

- POST /oauth2/token: Issues access tokens. Supported grant types include authorization_code, client_credentials, refresh_token, and urn:ietf:params:oauth:grant-type:jwt-bearer.
- POST /oauth2/revoke: Allows clients to programmatically revoke access or refresh tokens.
- POST /oauth2/introspect: Enables resource servers to validate the current state and associated scopes of an access token.
- **Decoupled Authentication APIs (Curity)**

The solution supports modern, cross-channel authentication patterns through standard extensions.

- **CIBA (**POST /bc-authorize**)**: Implements Client-Initiated Backchannel Authentication. A client initiates an authentication request for a known user, resulting in a push notification to the user's registered device. Key parameters include login_hint, binding_message, and client_notification_token. The response contains an auth_req_id for tracking.
- **Device Flow (**POST /device**)**: Implements the OAuth 2.0 Device Authorization Grant for QR-based login. The response includes a device_code, user_code, and verification_uri used to display a QR code.
- **Risk Engine APIs**

These internal APIs enable adaptive, risk-based step-up authentication.

- POST /api/v1/risk/evaluate: Called by the RAS Service after obtaining a risk score from the Fraud Management System. It evaluates the score and context against active policies to determine the required authentication factors.
    - **Request:** Includes userId, sessionId, fmsScore, fmsAction, and transaction details.
    - **Response:** Returns a decisionId, requiredFactors (e.g., ["FIDO2"]), fallbackFactors, and a stepUpRequired flag.

- POST /api/v1/risk/decision/{id}/fulfill: Called after the user successfully completes the required step-up authentication. It transitions the decision to FULFILLED and logs the outcome.
- GET /api/v1/risk/decision/{id}: Allows clients to query the current status of a pending authentication decision.
- **Transaction Authorization APIs (RAS Service)**

These APIs orchestrate the secure, device-bound transaction signing flow using SmartOTP.

Get Server Session Key

- **Endpoint:** GET /ras/key/server-sessionkey
- **Purpose:** Establishes a secure session between the mobile SDK and the backend. It returns the server's ephemeral public keys for encryption (RSA) and signature verification (ECC), along with a unique transactionId.

Create Auth Request

- **Endpoint:** POST /ras/auth-request
- **Purpose:** Called by the bank's backend to initiate a transaction authorization. The request includes the full transaction details, which are stored and linked to a sessionId.

Get Auth Request Details

- **Endpoint:** GET /ras/auth-request/{sessionId}
- **Purpose:** Called by the mobile SDK to fetch the transaction details for display and user confirmation. The response is signed and encrypted using the keys from the secure session.

Validate Transaction OTP

- **Endpoint:** POST /ras/transaction/validate-otp
- **Purpose:** Validates the TOTP/OCRA code entered by the user. A successful validation finalizes the transaction authorization.

Validate Basic OTP

- **Endpoint:** POST /ras/validate-basic-otp
- **Purpose:** Provides a general-purpose OTP validation endpoint for SMS or Email OTPs, used in step-up authentication or registration flows.

- **4.3.3 Flow Integration**

A typical transaction authorization flow using these APIs proceeds as follows:

1. **Bank Backend** → POST /ras/auth-request (creates pending authorization).
2. **Mobile SDK** → GET /ras/key/server-sessionkey (establishes secure session).
3. **Mobile SDK** → GET /ras/auth-request/{sessionId} (retrieves transaction details).
4. User reviews details and provides SmartOTP code.
5. **Mobile SDK** → POST /ras/transaction/validate-otp (validates OTP).
6. **RAS Service** (internally) → POST /api/v1/risk/evaluate and /fulfill (logs risk decision).
7. **Bank Backend** receives callback with final status.

This orchestrated flow ensures that every transaction is bound to a specific device, a validated user session, and a cryptographically verified OTP, meeting the requirements of **Circular 50/2024/TT-NHNN**.