**API Documentation for Customer IVR System (Arab Bank)**

## Introduction

This documentation outlines the RESTful APIs developed for Arab Bank’s IVR system integration. The APIs are designed to manage customer statuses, PIN operations, and customer subscriptions securely. All data exchanges, particularly for sensitive information such as PINs, are encrypted using **RSA encryption methods** through the **internal customer encryption API** to ensure data protection. Each API is structured with a clear request, response format, and error handling explanations.

## Security Protocol: HTTPS and Basic Authentication

**HTTPS**

To secure the data transmitted between the client and the server, these APIs are only accessible over HTTPS. HTTPS (Hypertext Transfer Protocol Secure) uses SSL/TLS encryption to protect data from interception during transit.

**Basic Authentication**

These APIs require Basic Authentication for access. Clients must include an Authorization header with each request, containing the base64-encoded credentials.

* **Credentials:** The client must use a valid username and password as configured in the system. These credentials are stored securely in the AuthConfig.js file.
* **Authorization Failure:** If the authorization header is missing, invalid, or contains incorrect credentials, the API will respond with a 401 Unauthorized error.

**Example of Adding Authorization Header**

Here’s an example of how to add the Authorization header using cURL for an HTTPS

request:

curl -X GET https://localhost:7999/api/customer/status/005494 \ -H "Authorization: Basic

$(echo -n 'root:Rainbow321#' | base64)"

**Error Responses for Authentication**

* **401:** Unauthorized

{ "message": "Authentication required" }

## Check Customer Subscription Status

**Description**

This API is used to check the subscription status of a customer based on their account number. It provides details on whether the customer's account is locked, dormant, or requires a PIN reset.

**Request URL**

GET /api/customer/status/:customerID

**Method**

#### GET

**Request Parameters**

* **customerID**(required): The Customer ID of the customer whose IVR status is being checked.

**Response**

* **customerID**: The customer’s ID number.
* **isLocked**: (boolean) Indicates if the account is locked due to multiple incorrect PIN attempts.
* **isDormant**: (boolean) Indicates if the account is dormant due to inactivity.
* **isRequiredReset**: (boolean) Indicates if the account requires a PIN reset.
* **status**: A string summarizing the overall status (e.g., "Active", "Locked", "Dormant").
* **isMigrated**: Indicates whether the account has been migrated (“1”, “0”)

**Sample Request**

GET https://localhost:7999/api/customer/status/005494

**Sample Response**

{

"customerID": "2001005494", "isLocked": false, "isDormant": false, "isRequiredReset": false, "status": Active,

“isMigrated”: 1

}

**Error Responses**

* **404**: Customer not found

{

"message": "Customer not found"

}

This error occurs when the account number provided does not exist in the database.

* **500**: Server error

{

"message": "Server error"

}

This error occurs if there is an issue with the database connection or internal server error.

## Delete Customer Service and Corresponding TIN

**Description**

This API allows the deletion of a customer’s IVR service and associated TIN. It is typically used when the customer no longer requires access to IVR services.

**Request URL**

DELETE /api/customer/delete/:customerID

**Method**

#### DELETE

**Request Parameters**

* **customerID**(required): The Customer ID of the customer whose service is to be deleted.

**Response**

* **message**: A success message indicating the customer's service and TIN have been deleted.

**Sample Request**

DELETE https://localhost:7999/api/customer/delete/005494

**Sample Response**

{

"message": "Customer service and TIN deleted successfully"

}

**Error Responses**

* **404**: Customer not found

{

"message": "Customer not found"

}

This error occurs if the account number provided does not match any customer records in the database.

* **500**: Server error

{

"message": "Server error"

}

This error occurs if the system encounters an issue during the transaction or database operation.

## View Customer PIN (TIN)

**Description**

This API retrieves the customer’s PIN (TIN), decrypts it internally, and re-encrypts it using RSA encryption with a public key generated through the internal customer encryption API. The encrypted PIN and the Key ID (KID) are returned to Arab Bank, allowing the bank to decrypt the PIN using the corresponding private key.

**Request URL**

GET /api/customer/pin/:customerID

**Method**

#### GET

**Request Parameters**

* **customerID**(required): The Customer ID of the customer whose PIN is being retrieved.

**Response**

* **customerID**: The customer’s ID number.
* **isMigrated:** Indicates whether the account has been migrated if No the **encryptedPIN and kid** will be empty**.**
* **encryptedPIN**: The PIN encrypted using RSA encryption with the generated public key.
* **kid**: The Key ID corresponding to the RSA key pair used for encryption. This KID should be used by Arab Bank to retrieve the private key for decryption.

**Sample Request**

GET https://localhost:7999/api/customer/pin/005494

**Sample Response**

{

"accountNumber": "005494",

"encryptedPIN": "f008953e0d3ca9680d526ee220bb9e6e, "kid": "f0abde57-5a51-494f-8d9f-5b9a259c964e",

"isMigrated": "No",

}

**Error Responses**

* **404**: TIN not found for the customer

{

"message": "TIN not found for the customer"

}

This error occurs when the account number is valid, but no TIN exists for the customer.

* **500**: Server error

{

"message": "Server error"

}

This error occurs if there is an issue retrieving or decrypting the TIN.

## Change Customer PIN (TIN)

**Description**

This API allows for updating the customer’s PIN (TIN). The new PIN must be encrypted using RSA encryption with a public key generated through the internal customer encryption API before being sent to the API. The API will decrypt the encrypted PIN using the corresponding private key and then store it securely after internal encryption.

**Request URL**

PUT /api/customer/pin/change

**Method**

#### PUT

**Request Body**

* **customerID**(required): The customerID number of the customer whose PIN is being updated.
* encryptedPIN (required): The PIN encrypted using RSA encryption with the public key.
* kid (required): The Key ID corresponding to the RSA key pair used for encryption.

**Response**

* **message**: A confirmation message that the PIN has been updated.

**Sample Request**

PUT https://localhost:7999/api/customer/pin/change

{

"customerID": "2001005494",

"encryptedPIN": "f008953e0d3ca9680d526ee220bb9e6e, "kid": "f0abde57-5a51-494f-8d9f-5b9a259c964e"

}

**Sample Response**

{

"message": "PIN updated successfully",

}

**Error Responses**

* **404**: Customer not found

{

"message": "Customer not found"

}

This error occurs if the account number provided does not exist in the database.

* **400:** New PIN must not be the same as the old PIN

{

"message": "New PIN must not be the same as the old PIN."

}

### 400: Invalid PIN

{

"message": "PIN must be exactly 4 digits."

}

* **500**: Failed to encrypt the new PIN

{

"message": "Failed to encrypt the new PIN"

}

This error occurs if the encryption process for the new PIN fails.

* **500**: Server error

{

"message": "Server error"

}

This error occurs if there is a database issue while updating the PIN.

## Subscribe Customer API

**Description**

This API allows for subscribing a new customer into the IVR system. The customer details are recorded in the database, and the PIN is encrypted and stored. The PIN must be encrypted using RSA encryption with a public key generated through the internal customer encryption API before being sent to the API.

**Request URL**

POST /api/customer/subscribe

**Method**

#### POST

**Request Body**

* **accountNumber** (required): The customer’s account number.
* **branchID** (required): The customer’s branch ID.
* **country** (required): The customer’s country.
* **mobileNumber** (required): The customer’s mobile number.
* encryptedPIN (required): The PIN encrypted using RSA encryption with the public key.
* **kid** (required): The Key ID corresponding to the RSA key pair used for encryption.

**Response**

* **message**: A success message indicating the customer was registered.

**Sample Request**

**URL :**

POST https://localhost:7999/api/customer/subscribe

**Body JSON :**

{

"accountNumber": "005494", "branchID": "201",

"country": "1",

"mobileNumber": "9613115914",

"encryptedPIN": "f008953e0d3ca9680d526ee220bb9e6e, "kid": "f0abde57-5a51-494f-8d9f-5b9a259c964e"

}

**Sample Response**

{

"message": "Customer registered successfully",

}

**Error Responses**

* **400**: Customer already exists

{

"message": "Customer already exists in the IVR system"

}

This error occurs if the customer is already present in the system.

* **400:** Invalid PIN

{

"message": "PIN must be exactly 4 digits."

}

* **500**: Failed to encrypt the PIN

{

"message": "Failed to encrypt the PIN"

}

This error occurs if there is an issue encrypting the provided PIN.

* **500**: Server error

{

"message": "Server error"

}

This error occurs if there is an issue with the database transaction.

## Update Customer Account Status

**Description**

This API allows for updating a customer’s account status by toggling either the IsDormant or IsLocked field based on the specified newValue. If the status is already set to the desired value, a message indicates no change is needed.

**Request URL**

PUT /api/customer/status/update

**Method**

#### PUT

**Request Parameters**

* **Body JSON**:
  + customerID(required): The customerID of the customer whose status is being updated.
  + fieldToUpdate (required): Specifies either "IsDormant" or "IsLocked"

as the field to be updated.

* + newValue (required): A value of 1 (active) or 0 (inactive) to set the specified field.

**Response**

* message: Provides a message based on the status update result:
  + If the field was updated, the response will confirm the change.
  + If the field was already set to the requested value, a message indicates no change was necessary.

**Sample Request**

PUT https://localhost:7999/api/customer/status/update

Body JSON:

{

"customerID": "201003608", "fieldToUpdate": "IsDormant", "newValue": 1

}

### Sample Responses

* **Success: Account status updated**

{

"message": "Account has been successfully set to dormant"

}

* **Success: Account already has requested status**

{

"message": "Account is already dormant"

}

### Error Responses

* **400: Invalid field specified**

{

"message": "Invalid field specified. Choose either IsDormant or IsLocked."

}

Occurs if the fieldToUpdate parameter is not "IsDormant" or "IsLocked" or if

newValue is not 0 or 1.

### 404: Customer not found

{

"message": "Customer not found"

}

Occurs if the accountNumber provided does not exist in the database.

### 500: Server error

{

"message": "Server error"

}

Occurs if there is an issue with the database connection or internal server error.

**Notes on RSA Encryption and Key Management**

**AES-GCM (Internal)**

* Salt: 16 byte
* IV: 12 bytes
* Key Derivation: PBKDF2 HMAC-SHA256, 100k iter, 256-bit key
* Tag: 16 bytes

**RSA (Transmission)**

* Padding: PKCS#1 v1.5
* Base64URL

**GCP RSA Key API encode ciphertext**

Generate Key Pair:

1. POST https://api-int.gcptest.arabbank.plc/digital-banking/ms/security/key-generator/v2/rsa/keys

2. Headers:

3. accept: application/json

4. channel-name: IVR

5. Content-Type: application/x-www-form-urlencoded

6. client\_id: YOUR\_CLIENT\_ID

7. Body (form):

8. country=JO

9. scope=Helios

Response:

1. {

2. "data": {

3. "kid": "<Key ID>",

4. "kty": "RSA",

5. "use": "sig",

6. "alg": "RS256",

7. "e": "<Base64URL exponent>",

8. "n": "<Base64URL modulus>"

9. }

Encrypting the PIN:

1. const forge = require('node-forge');

2. const base64url = require('base64url');

3.

4. function encryptPINWithPublicKey(pin, n\_b64, e\_b64) {

5. const BigInteger = forge.jsbn.BigInteger;

6. const n = new BigInteger(base64url.toBuffer(n\_b64).toString('hex'), 16);

7. const e = new BigInteger(base64url.toBuffer(e\_b64).toString('hex'), 16);

8. const publicKey = forge.pki.setRsaPublicKey(n, e);

9. const raw = publicKey.encrypt(pin, 'RSAES-PKCS1-V1\_5');

10. return base64url.fromBase64(forge.util.encode64(raw));

11. }

Retrieving the Private Key & Decrypting (at Arab Bank):

1. const fetch = require('node-fetch');

2. const forge = require('node-forge');

3. const base64url = require('base64url');

4.

5. // 1) Fetch private key components (n, e, d)

6. async function getPrivateKey(kid) {

7. const resp = await fetch(

8. `https://api-int.gcptest.arabbank.plc/digital-banking/ms/security/key-generator/v2/rsa/keys/${kid}?country=JO&scope=Helios`,

9. { method: 'GET', headers: {

10. accept: 'application/json',

11. 'channel-name': 'IVR',

12. client\_id: 'YOUR\_CLIENT\_ID'

13. }}

14. );

15. const { data } = await resp.json();

16. return data;

17. }

18.

19. // 2) Decrypt incoming encryptedPIN

20. async function decryptPIN(encryptedPIN, kid) {

21. const { n: n\_b64, e: e\_b64, d: d\_b64 } = await getPrivateKey(kid);

22. const toBigInt = b => new forge.jsbn.BigInteger(base64url.toBuffer(b).toString('hex'), 16);

23. const privateKey = forge.pki.setRsaPrivateKey(toBigInt(n\_b64), toBigInt(e\_b64), toBigInt(d\_b64));

24. const cipherBytes = forge.util.decode64(base64url.toBase64(encryptedPIN));

25. return privateKey.decrypt(cipherBytes, 'RSAES-PKCS1-V1\_5');

26. }

27.

28. // Example

29. decryptPIN('Base64URLCipherHere', 'your-kid-here')

30. .then(pin => console.log('Decrypted PIN:', pin))

31. .catch(console.error);

32.

**Security Considerations**

* Never log or expose decrypted PINs or private key material.
* Store all keys in secure vaults with access control.
* Return user-friendly errors; avoid leaking implementation details.