# **DPIP Integration Guide**

## **1.0 System Overview**

Our system uses a hybrid architecture to balance high-speed, real-time screening with other essential workflows. Your integration will target two different services depending on the use case.

* **For Real-Time Screening**: Your core system will call the **Bank Edge Service** deployed in your local infrastructure.
* **For All Other Use Cases**: Your core system will call the **Central DPIP Service** directly for workflows like data ingestion, validation, and dispute resolution.

### **1.1 Key Components**

* **Partner's Core System**: Your primary transactional or core banking system. It will initiate all API calls.
* **Bank Edge Service (For Screening Only)**: A service deployed within your bank's infrastructure, specifically designed and optimized for high-speed, local screening. It is not used for any other purpose.
  + **Screening Service**: The API endpoint your system calls for real-time fraud checks.
  + **Replication Service**: A background process that keeps the local fraud data synchronized by fetching updates from the Central DPIP Service.
* **Central DPIP Service (For Ingestion, Validation, etc.)**: The central, authoritative service that we host. It handles all non-screening workflows and serves as the source of truth for the entire network.

## **2.0 Getting Started: Onboarding & Setup**

Before you can begin integration, your team will need to configure your development environment to connect to our services and ensure secure communication.

### **2.1 Environment Setup**

We provide two distinct environments: a **Sandbox** for development and testing, and a **Production** environment for live traffic. You must begin your development against the Sandbox environment.

**Environment URLs**

Your integration will need to target different URLs based on the workflow.

| **Workflow** | **Target Service** | **Environment** | **URL** |
| --- | --- | --- | --- |
| **Screening** (/check) | Bank Edge Service | Sandbox | http://<your-bank-edge-host> |
| Production | http://<your-bank-edge-host> |
| **Ingestion,Augmentation,Disputes,Feedback** | Central DPIP Service | Sandbox | https://sandbox.central.dpip.service |
| Production | https://central.dpip.service |

### **2.2 IP Whitelisting**

For security, all communication must originate from whitelisted IP addresses. Provide the public IPs your **Core System** will use to call the **Central DPIP Service**. Please contact our support team with lists of IPs to begin this process.

## **3.0 Integration Paths**

You have two options for integrating with our services. We strongly recommend using the provided SDK as it simplifies development and handles complex security requirements automatically.

### 

### **3.1 Path A: SDK Integration (Recommended)**

#### 

#### **3.1.1 Why Use the SDK?**

The SDK is designed to accelerate your development process by handling many complex and repetitive tasks automatically. Key benefits include:

* **Automated Data Handling**: Automatically handles data standardisation, normalisation, and the hashing of PII.
* **Simplified Security**: Abstracts away the low-level details of manual request signing (JWS)
* **Increased Focus**: Allows your team to focus on core business logic instead of complex security protocols and data prep.
* **Faster Development**: Reduces integration time and effort significantly.

#### 

#### **3.1.2 Supported Languages**

Currently, we provide an official SDK for **Java**.

To access the SDK, here is the link to the official SDK for Java language: <https://drive.google.com/drive/folders/1nhzYiSG4uiDYCUPB8bkgp2yFdMfrtofd?usp=sharing>

#### 

#### **3.1.3 Installation & Dependencies**

To incorporate the SDK into your Java project, add the following dependency to your pom.xml file:

| <dependency>  <groupId>com.dpip</groupId>  <artifactId>assembly</artifactId>  <version>0.0.1</version>  <scope>system</scope>  <systemPath>  ${project.basedir}/path/of/jar/file.jar  </systemPath> </dependency> |
| --- |

#### 

#### **3.1.4 Configuration**

To use the SDK, you will need to initialize the appropriate service client with the target URL for the workflow you are implementing (e.g., the Central DPIP Service URL for ingestion).

| // Example: Initializing the service for data ingestion  // Import ingestionsdk import com.dpip.ingestionsdk.services.IngestionService;  String centralApiUrl = "https://sandbox.central.dpip.service"; // Use the correct environment URL String reportFormat = "json"; // or "xml" String edgeEd25519PrivateKey = "/path/of/ed25519/privet/key.pem" // or data in .pem file  String edgeRsaPrivateKey = "/path/of/rsa/privet/key.pem" // or data in .pem file  String bankName = "MyBank name";  // The 'IngestionService' handles all communication for reporting fraud  IngestionService ingestionService = new IngestionService(  hostnameCentral: centralApiUrl,   reportType: reportFormat,  edgeRsaPrivateKey: edgeRsaPrivateKey,  edgeEd25519PrivateKey: edgeEd25519PrivateKey,   bankName: bankName); |
| --- |

#### 

#### **3.1.5 Quick Start Example**

Here is a simple example of how to send a fraud report using the SDK's IngestionService. The SDK handles the conversion of the JSON file into the required RegisterRequest object and all security protocols.

| try {  // Provide the path to your JSON report file  String reportFilePath = "path/to/your/report.json";  String centerEd25519PublicKey = "/path/of/ed25519/public/key.pem" // or data in .pem file  String centerRsaPublicKey = "/path/of/rsa/public/key.pem" // or data in .pem file   ingestionService.registerReportFromJson(reportFilePath);  ingestionService.registerReportFromJson(  reportFilePath,  centralsRsaPublicKey,  centerEd25519PublicKey  );  System.out.println("Fraud report sent successfully!");  } catch (Exception e) {  // Handle potential exceptions, e.g., file not found, network issues  e.printStackTrace(); } |
| --- |

### 

### **3.2 Path B: Direct API Integration**

#### **3.2.1 When to Use the API**

This method provides maximum flexibility and is intended for teams that need to build their own integration logic or are using a programming language for which an SDK is not available.

#### 

#### **3.2.2 Core Requirements**

If you choose this path, your team is responsible for implementing all aspects of the communication, including:

* **Data Standardization**: Ensuring all data conforms to the required formats.
* **Identifier Hashing**: Hashing all PII according to the specified standards.
* **Request Signing (JWS)**: Signing the request to ensure its integrity.
* **Payload Encoding** : Encode the entire request body.

#### 

#### **3.2.3 Manual Request Building**

Building a request manually involves a multi-step process of data preparation, hashing, encryption, and signing before making the final HTTP call. Detailed instructions for this process are covered in the **Authentication & Security** section.

## **4.0 Authentication & Security**

### 

### **4.1 Security Overview**

To ensure end-to-end security, the DPIP platform integration employs a digital signature model for all API communication. This approach guarantees the **integrity** and **authenticity** of every transaction.

The process involves signing the request payload using a JSON Web Signature (JWS). This ensures that the message content can be verified for its integrity and origin, providing robust protection against various attack vectors like message tampering.

In addition to payload-level integrity, all network communication is protected using mutual TLS (mTLS). This provides **confidentiality** by establishing a trusted and encrypted transport channel, where both the client and server authenticate each other using digital certificates.

Banks not utilizing the provided SDK must adhere to the detailed cryptographic procedures for JWS signing outlined below for every API call.

### **4.2 Procedure**

| **Step 1: PII Hashing**  Before any cryptographic operations are performed, the raw JSON payload must be prepared. A critical security measure is the irreversible hashing of all Personally Identifiable Information (PII).   * **PII Hashing**: All sensitive identifiers (e.g., PAN, mobile numbers) must be hashed. * **Algorithm**: **SHA-256** * **Process**: The identifier is hashed directly. The formula is: hashed\_identifier = SHA-256 (identifier) |
| --- |

### 

| **Step 2: Payload Signing with JWS**  After preparing the payload, it is signed to create a JWS token. This step ensures data integrity and proves the origin of the request.   * **JWS Header**: The header must specify the signing algorithm and the key identifier.   + alg (Algorithm): **EdDSA**   + kid (Key ID): The bank's unique signing key identifier. * **Signature**: The stringified payload is signed using the bank's private key corresponding to the kid. * **Output**: The result is a compact JWS token in the HEADER.PAYLOAD.SIGNATURE format. |
| --- |

#### 

| **Step 3: Transmit the Request**  The JWS string serves as the entire body of the API request.   * **Endpoint**: POST /register * **Host**: sandbox.central.dpip.service * **Header**:   + Content-Type: application/jws * **Request Body**: The payload is the complete JWS string, structured as follows:   {  "header": BASE64URL(HEADER),  "payload": BASE64URL(PAYLOAD),  "signature": BASE64URL(SIGNATURE)  }   * **Response Body**: The payload is the complete JWS string, structured as follows:   {  "header": BASE64URL(HEADER),  "payload": BASE64URL(PAYLOAD),  "signature": BASE64URL(SIGNATURE)  } |
| --- |

## 

## **5.0 Core Workflows & API Reference**

### **5.1 Data Enrichment Flow**

**Description**

API for augmenting the central registry with more information about the reported identifier

**Endpoint**

Sample bank webhook endpoint : Post /augment/ask

**Request Structure**

| **Field** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| tracker\_id | String | Yes | The unique identifier corresponding to each augmentation request. |
| id\_value | String | Yes | The value of the identifier. |
| id\_type | String | Yes | The type of the identifier (AADHAR, PAN, MOBILE, EMAIL, UPI\_ID, BANK\_ACCOUNT\_NUMBER, IFSC\_CODE, IP\_ADDRESS, ACCOUNT\_HOLDER\_NAME, BANK\_NAME, GST\_NUMBER, MAC\_ADDRESS, DRIVING\_LICENSE, UDYAM\_REGISTRATION, NPR\_REFERENCE\_NUMBER, VOTER\_ID, PASSPORT, NREGA\_CARD\_NUMBER, CIN, TAN, LLPIN, ABHAID). |

**Sample Request**

| {  "tracker\_id": "abc-12345",  "id\_value": "abc",  "id\_type": "AADHAAR" } |
| --- |

### 

**Acknowledgement Sample**

| { "status":"REQUEST\_RECEIVED", "tracker\_id":"abc-12345" } |
| --- |

**Augmentation callback Structure**

**Endpoint**

Post /augment/submit

| **Field** | **Type** | **Required** | **Description** |
| --- | --- | --- | --- |
| tracker\_id | String | No | The unique identifier corresponding to each augmentation flow. This field is only needed when responding to an augmentation request sent by the Center. |
| entity\_id\_value | String | Yes | The value of the primary entity identifier(e.g. Customer ID) associated with the input ID. |
| entity\_id\_type | String | Yes | The type of the primary entity identifier (e.g., "CUSTOMER\_ID"). |
| augmented\_ids | Array | Yes | An array of objects, where each object represents an associated identifier. Length of the array > 0 |
| augmented\_ids.type | String | Yes | The type of the augmented identifier. |
| augmented\_ids.value | String | Yes | The value of the augmented identifier. |

Examples

**Sample Augmentation Callback structure**

**Use case 1:** Associated identifiers are available

| {  "tracker\_id": "abc-12345",  "entity\_id\_value": "1234567890",  "entity\_id\_type": "CUSTOMER\_ID",  "augmented\_ids": [  {  "type": "MOBILE",  "value": "CAEAC981C9F2F7F057850AF53E0E70F2C1C48331698492078870D96320AF5133"  },  {  "type": "PAN",  "value": "F094BD9BAFB42A1BC4B026ABCF60B68FB32E1CF3509F9FAF6F71F0F93272F063"  }  ] } |
| --- |

### 

Use case 2 : New identifiers are added by the customer

| {  "entity\_id\_value": "1234567890",  "entity\_id\_type": "CUSTOMER\_ID",  "augmented\_ids": [  {  "type": "MOBILE",  "value": "20CBB07F09FAA38572127E89A847BE0D70979C7F24CB47D49B120A11E0D30644"  },  {  "type": "PAN",  "value": "128C0ECBF9E37F52DDE2040A6DD6A6A1E5CA71E59B27EE24D2B93FDC2BC5F8DC"  }  ] } |
| --- |

### 

## **6.0 Annexure**

### **6.0.1 Overall Error codes & Reason**

This section provides details on the HTTP status codes you may encounter when integrating with our APIs.

| **Status Code** | **Meaning** | **Possible Causes & Resolution** |
| --- | --- | --- |
| 200 - OK | Success | The request was successfully received and processed. The response body will contain the requested data. |
| 400- Bad Request | Client Error | The server could not process the request due to an issue on the client side. Check for malformed JSON in the request body. Ensure all required fields are present. Verify that enum values (e.g., id\_type, report\_type) are correct. |
| 401-Unauthorized | Authentication Error | The request lacks valid authentication credentials. Ensure the JWS security flow was followed correctly. Verify that the JWS signature is valid and has not expired. Confirm you are using the correct keys for signing and encryption. |
| 403 Forbidden | Permission Denied | You are authenticated, but you do not have permission to perform the requested action. Check if your organization is enabled for the specific workflow (e.g., Validation). |
| 404 Not Found | Not Found | The requested API endpoint does not exist. Check the URL for typos. Ensure you are targeting the correct service (Bank Edge vs. Central DPIP) for the workflow. |
| 429 Too Many Requests | Rate Limit Exceeded | You have sent too many requests in a given amount of time. Implement a retry mechanism with exponential backoff and jitter |
| 500 Internal Server Error | Server Error | An unexpected error occurred on our server. This is a generic server error. Wait a few moments and then retry the request using an exponential backoff strategy. If the problem persists, check our status page and contact DPIP with the request ID if available. |
| 503 Service Unavailable | Service Unavailable | The server is temporarily unable to handle the request, often due to maintenance or overload. Wait and retry the request after a delay. Check our status page for any announced maintenance windows. |

### **6.0.2 Identifier Types**

| **Identifier Type** | **Description** | **Type** | **Sample Value** |
| --- | --- | --- | --- |
| AADHAR | A 12-digit unique identification number issued by the UIDAI to residents of India. | string | 4321 5678 9012 |
| PAN | A ten-character alphanumeric identifier issued by the Indian Income Tax Department. | string | GHIJK5678L |
| MOBILE | A 10-digit number in India, starting with 6, 7, 8, or 9. | string | 9988776655 |
| EMAIL | A unique identifier for sending and receiving electronic mail. | string | sample.user@example.com |
| UPI\_ID | A unique identifier used for digital payments in India, e.g., 'username@bankname'. | string | john.doe@okhdfcbank |
| BANK\_ACCOUNT\_NUMBER | A unique identifier for a bank account, ranging from 9 to 18 digits. | string | 0987654321098 |
| IFSC\_CODE | An 11-character alphanumeric code that uniquely identifies a bank branch in India. | string | SBIN0000123 |
| IP\_ADDRESS | A unique identifier for a device on a network, consisting of four numbers separated by periods. | string | 203.0.113.1 |
| ACCOUNT\_HOLDER\_NAME | Name of the bank account holder. | string | Priya Sharma |
| BANK\_NAME | Name of the bank or financial institution. | string | ICICI |
| GST\_NUMBER | A 15-character unique identifier for businesses registered under the GST regime in India. | string | 29GHIJK5678L1Z9 |
| MAC\_ADDRESS | A unique identifier for network interfaces, represented as six groups of two hexadecimal digits. | string | 001A2B3C4D5E |
| DRIVING\_LICENSE | An authorization to operate a motor vehicle, typically two letters followed by 13 digits. | string | KA5120240012345 |
| UDYAM\_REGISTRATION | A unique identification number for small and medium enterprises in India. | string | UDYAMKA03001234 |
| NPR\_REFERENCE\_NUMBER | A unique identifier for individuals in the National Population Register, consisting of 'NPR' and 9 digits. | string | NPR987654321 |
| VOTER\_ID | A unique ID issued by the Election Commission of India to eligible voters. | string | WXY9876543 |
| PASSPORT | An official document certifying identity and citizenship, typically one letter followed by seven digits. | string | P1234567 |
| NREGA\_CARD\_NUMBER | A unique identifier for individuals registered under the MGNREGA in India. | string | KA050011239876 |

### 