This is a guide for gpt model which will contain all the information from eko's website, so that the model can answer the queries of the users.

Eko for Developers: Welcome to the Eko Developer Portal! Eko offers a suite of services, APIs, and tools for developers, enabling you to process small value financial transactions via IMPS/NEFT, verify bank account details instantly, enable cash collection for third-party applications, offer biometric-based cash-out via AePS, collaborate with lending organizations, and simplify KYC with PAN card verification. Our RESTful APIs provide easy integration, with all responses in JSON format. Rest assured, our robust security solutions allow secure interactions, even from client-side websites or applications, without exposing your secret-key.

AUTHENTICATION:
1)**API Key:** Authenticate your API requests using a developer_key, secret-key and secret-keytimestamp which should be kept confidential
2)For UAT, a dummy **developer_key** can be used from platform credentials section(which will be common for everyones's testing purpose), while for Production a user-specific developer-key and auth-key(which will be used in the code to generate secret-key and secret-key timestamp) will be provided when a user mails to sales.engineer@eko.co.in that his, UAT testing is done and he wants Live credentials for production.
3)How to generate the secret-key?:1:Encode key using base64 encoding technique.2:Generate current timestamp (in milliseconds since UNIX epoch), i.e. secret-key-timestamp (Check currentmillis.com to understand the timestamp format).3:Compute the signature by hashing salt and base64 encoded key using Hash-based message authentication code HMAC and SHA256.4:Encode the signature using base64 encoding technique and use this as secret-key.Following are the codes:
PHP:

```
<?php

// Initializing key in some variable. You will receive this key from Eko via email
$key = "d2fe1d99-6298-4af2-8cc5-d97dcf46df30";

// Encode it using base64
$encodedKey = base64_encode($key);

// Get current timestamp in milliseconds since UNIX epoch as STRING
// Check out https://currentmillis.com to understand the timestamp format
$secret_key_timestamp = (int)(round(microtime(true) * 1000));

// Computes the signature by hashing the salt with the encoded key
$signature = hash_hmac('SHA256', $secret_key_timestamp, $encodedKey, true);

// Encode it using base64
$secret_key = base64_encode($signature);

echo $secret_key;
?>
```

Code in Python:

```
import hmac
import base64
import hashlib
import time

secret_key_timestamp = str(int(round(time.time()*1000)))
key = 'd2fe1d99-6298-4af2-8cc5-d97dcf46df30'
dig = hmac.new(
  base64.b64encode(key), secret_key_timestamp, hashlib.sha256
).digest()
secret_key = base64.b64encode(dig).decode()
```

Notes:
-Refer to following link for infor: https://developers.eko.in/docs/authentication
-Only IP which is in India will be whitelisted while going on the production mode. IP which is present outside India will not be whitelisted as per compliance
-developer_key for the Staging Environment: becbbce45f79c6f5109f848acd540567
-key for secret-key and secret-key-timestamp(auth-key): d2fe1d99-6298-4af2-8cc5-d97dcf46df30.The secret-key and secret-key-timestamp have to be generated dynamically. Refer to the link https://developers.eko.in/docs/authentication for the dynamic secret-key and secret-key-timestamp generation.
-initiator_id - 9962981729. user_code - 20810200. The following initiator_id and user_code are dummy for staging enviornment. For production enviornment you will get you own exclusive initiator_id and user_code.
//
Eko's API equips you with a wide array of capabilities: 1) Create customers, 2) Inquire about customer profiles, 3) Add entities like recipient bank accounts to customer profiles, 4) Process financial transactions on behalf of customers, including IMPS/NEFT, 5) Enable third-party lending organizations to provide working capital loans to Eko's network of merchants, distributors, channel partners, and customers, 6) Collect repayment of loans, 7) Enable verification of bank account details, 8) Facilitate the sale of third-party products and services via the Eko merchant network, 9) Verify your customers and merchants via eKYC, 10) Process biometric-based cash-out transactions through AePS (Aadhaar-enabled payment system), and 11) Verify your customer's PAN card for KYC
//

- Customer Entity:
- id => Data Type: string. Description: Value of customer ID.
- customer_id_type => Data Type: string. Description: Type of customer ID.
- state => Data Type: integer. Description: Signifies the current state of the customer.
- state_desc => Data Type: string. Description: Description of the current state of the customer.
- limit => Data Type: array. Description: Signifies the amount available for the customer to transact in the month.
- balance => Data Type: double. Description: Current balance of the customer.
- name => Data Type: string. Description: Name of the customer.

- Recipient Entity:

- id => Data Type: string. Description: ID of the recipient; needs to have the same format as mentioned in recipient_id_type.
- recipient_id_type => Data Type: string. Description: Can have 2 values: 1. acc_ifsc, 2. acc_bankcode.
- recipient_id => Data Type: string. Description: Unique ID of the recipient for the customer; this will be used in the transaction API.
- name => Data Type: string. Description: Name of the recipient.
- recipient_mobile => Data Type: integer. Description: Mobile number of the recipient.
-
- Transactions Entity:
- tid => Data Type: integer. Description: Unique transaction ID on Eko platform.
- client_ref_id => Data Type: string. Description: Unique tid on partner platform.
- timestamp => Data Type: Description: Time at which transaction was done, should be in TZ format.
- currency => Data Type: string. Description: Currency in which transaction is being processed.
- state => Data Type: integer. Description: Signifies the state of the transaction.
- channel => Data Type: integer. Description: Signifies the mode of transaction.
- refund_tid => Data Type: integer. Description: Unique transaction id on Eko platform when a refund is done.
-
- Bank Entity:
- bank_code => Data Type: string. Description: Short code for each bank.
- name => Data Type: string. Description: Name of the bank.
- isverificationavailable => Data Type: boolean. Description: Signifies availability of account name verification.
- ifsc_status => Data Type: number. Description: Signifies if IFSC is required for addition of bank account number.
- code => Data Type: string. Description: Short name of the bank.

Some more attributes with their meaning are:

recipient_id : Unique Id generated while adding the receipient

amount : The amount value which customer needs to transfer.

timestamp: The current timestamp

currency: This will be a static value. value of parameter is INR

customer_id: Customer’s mobile number

initiator_id: The unique cell number with which partner is on-boarded on Eko's platform

client_ref_id : Unique reference number of partner's system, please make it as unique as possible so that it does not match with any other partner's unique reference id. (e.g. First 3 or 4 letters of your organisation + current timestamp)

hold_timeout: pass any static value (e.g. 10)

state: This will be a static value and value of this parameter will always be 1

channel: Money can be sent via 2 channels: IMPS or NEFT. (1 - NEFT, 2 - IMPS)

latlong: latlong of partner's retailer of whom merchant_document_id is passed Pass either of them.

->Transaction Status and Transaction descriptions.For all financial transactions, ***status = 0*** should be treated as successful else fail and the current state of the transaction can be retrieved from ***tx_status*** and ***txstatus_desc*** parameter.

-tx_status=0 , txstatus_desc=Success.

-tx_status=1 , txstatus_des=fail

-tx_status=2 , txstatus_desc=Response Awaited/Initiated (in case of NEFT).

-tx_status=3 , txstatus_desc=Refund Pending.

-tx_status=4 , txstatus_desc=Refunded.

-tx_status=5 , txstatus_desc=Hold (Transaction Inquiry Required).

//DMT Payment flow:Any financial transaction has the following flow:

DMT Api FLOW (Domestic Money Transfer)-

1)First you need to onboard your merchants/ retailers using the Onboard user API(https://developers.eko.in/reference/onboard-user).

2)Then you need to check that whether the customer is already existing or not on EKO's platform using the Get Customer Information API(https://developers.eko.in/reference/get-customer-info).

3)If the customer already exist, then fetch his already registered recipients using the Get List of Recipients API. If the customer is not registered, then create a customer using the Create Customer API(https://developers.eko.in/reference/create-customer).

4)After the creation of the customer you need to verify the customer using the Verify Customer Identity API(https://developers.eko.in/reference/verify-customer-identity). You will not receive an OTP on UAT. If you want to resend the OTP hit the Resend OTP API

5)If the customer is already registered and no recipient has been added then add the recipient using the Add Recipient API(https://developers.eko.in/reference/add-recipient). You can check the list of recipients added from the Get list of recipients API(https://developers.eko.in/reference/get-all-recipients)

6)Once the recipient has been added or the already registered recipient has been selected then you can send the cash using Initiate Transaction API(https://developers.eko.in/reference/initiate-transaction) via either IMPS or NEFT channel.

7)You can check the status of any transaction using the Transaction Inquiry API(https://developers.eko.in/reference/money-transfer-inquiry). If any transaction goes into the refund pending then you can process the refund using the Refund API(https://developers.eko.in/reference/refund).

AePs Api FLOW (Aadhar Enabled Payment System)

1. Firstly , Onboard your users (merchants) by initiating the **Onboard-User API**.

2. Activate AePS services for your users by initiating the **Activate-Service API**(service code-52 for Aeps(FINO), service_code-43 for FINGPAY AND GATEWAY, service_code=51 for AADHAR PAY) with user's unique code (returned in response to the Onboard-User API).
3. After activating the Aeps service, you have to do merchant's e-KYC. There are two ways to do ekyc.

   -1st:E-KYC Through FINGPAY: To do ekyc through FINGPAY, you will have to call 3apis consecutively which are:

   e-KYC OTP Request(https://developers.eko.in/reference/e-kyc-otp-request), e-KYC OTP Verification(https://developers.eko.in/reference/e-kyc-otp-verification), e-KYC using Biometric(https://developers.eko.in/reference/e-kyc-using-biometric), After hitting these 3apis yous E-KYC will be successfull through FINGPAY

   -2:E-KYC through FINO: To do e-kyc through FINO, you just have to call one api, i.e e-KYC using Biometric(https://developers.eko.in/reference/e-kyc-using-biometric)

4. After the merchant's E-KYC is complete(either through FINGPAY or FINO), he is required to do Daily Authentication(https://developers.eko.in/reference/aeps-daily-authentications)
5. After his these two steps i.e E-KYC and Daily Authentication is complete, the merchant can proceed towards doing a transaction.
6. Keep in mind, the merchant has to do Daily Aunthentication everyday while E-KYC he has to do only once for a merchant.
7. Also keep in mind E-KYC through FINO takes 2-3 days, while E-KYC through FINGPAY is instant(therefore e-kyc through FINGPAY is recommended)
8. Inquire about the status of your user on Eko platform by initiating the **User Services Enquiry** API

NOTE:

-For AePS Cash-out, withdrawal limit is : ₹ 10,000 per transaction . 5 transactions per Aadhaar per day

-For Aadhar Pay, withdrawl limit is : ₹ 10,000 per transaction

AePs Gateway:

-AePS Gateway definition: Eko has designed AePS Gateway to allow you to securely process a biometric-based cash-out, mini statement and balance inquiry services for your customers.i.e. Eko will provide you with the frontend application where the customer can perform these action ,partners can upload their logo in it but keep in mind that the url would be -***https://gateway.eko.in***

-To activate gate service use, service_code=43. (refer to: https://developers.eko.in/reference/activate-service)

-To integrate gateway refer to following links in the order:

https://developers.eko.in/docs/aeps-gateway-workflow,https://developers.eko.in/docs/aeps-web-integration,https://developers.eko.in/docs/aeps-backend-integration

COMMON ERRORS/FAQ FOR AePS GATEWAY:

Que)How can i integrate the AePS Gateway?

Ans)You can integrate the AePS Gateway in both Web and Android Mobile application :

AePS Gateway Web : Refer to the link : https://developers.eko.in/docs/aeps-web-integration

Que) I am getting "Error in authentication"

Ans) You might be getting this error when you open gateway, It might be because you haven't changed the value "environment": "uat" to "environment": "production" in aeps.config.

- Another reason could be due to incorrect secret-key or timestamp, Make sure that the secret-key and secret-key-timestamp must be generated dynamically and correctly.

-Also check your aeps.config values properly

Que) I am getting "Authentication Failed...Initiator access forbidden"

Ans) Make sure the AePs service (service_code=43) is activated for the user_code

Que) I am getting "Connection to callback server failed. Please try again"

Ans) This error comes when the CORS headers are not implemented properly on the callback URL they are passing in the code. You can refer to the link "https://developers.eko.in/docs/enable-cors" .

Que) I am getting "Connection to callback server failed. Please try again (Status = 200 OK)"

Ans) There must be some problem in your callback setup. Please check for the debit hook response in the network tab. The debit hook response must be passed from your callback URL and must be passed in the JSON format only with the correct parameter values.

Que)I am getting "Transaction verification timeout" error message

Ans)This happens when we are expecting the debit hook response from your end but we are not getting the same from your end or not getting in the JSON format. Check for the response in the network tab.

Que) I am getting "Authenticaton for secret-key failed / Authenticaton for request-hash failed"

Ans) Please check the generation of the secret-key/ request-hash from your end by printing each and every value, it should be generated with the correct key and as per with the mentioned steps in the documentation(https://developers.eko.in/docs/aeps-backend-integration)

Que) Is it necessary to pass the callback URL with HTTPS?

Ans) Yes, the callback URL must be HTTPS only if hosted on server otherwise the browser will give the CORS headers issue.Example Callback: 'https://your-website.com/eko_aeps_callback'

Following is the list of some General errors common to all apis

1) Error 403 or Error 403 forbidden

Solution: It is usually due to incorrect secret-key or timestamp from your end. Please check the generation code from the following link: https://developers.eko.in/docs/authentication . Also make sure to use you auth_key in generation code (and not use your developer key)

2)500 internal server error

Solution: It usually implies that the api is not able to make connection to our servers. In staging remove the port 25004 from the url. And in production re-check your URL and http method.

3)415 Unsupported Media Type

Solution: Check the Content-type of the api from its respective Eko page and check if you are using the same content-type or not.

4) No mapping rule matched

Solution: In the URL change v1 to v2 or vice-versa

5) 405: method not allowed

Solution: Check the http request of the api from the api's respective page and put the correct http request which could be Post, Put, Get

6)Merchant id incorrect or incorrect pin

Solution: Check his e-kyc status

7) Agent not allowed/ Agent not allowed to do this transaction

Solution: Activate the service for the merchant.

//-Payout Api Flow

-Step 1: Activate Service (Only in production) with service_code = 45. Refer the following link https://developers.eko.in/reference/activate-service-aeps-copy.

Only acceptable url for Activate Service api is:

(Method:PUT) https://staging.eko.in/ekoapi/v1/user/service/activate.

For production the URL would be:

(Method:PUT) https://api.eko.in/ekoicici/v1/user/service/activate

Please check your URL carefully from the api page so as to not get any errors

-Step 2 : Use the Initiate Payout api to, Initiate a fund transfer to any bank account. Refer the following link:
https://developers.eko.in/reference/initiate-fund-transfer

Only acceptable url for Initiate Payout api is :

(Method:POST) https://staging.eko.in/ekoapi/v1/agent/user_code:{user_code}/settlement

Production url for the same would be like:

(Method:POST) https://api.eko.in/ekoicici/v1/agent/user_code:{user_code}/settlement

Step 3: Send us your callback to configure So that using that callback you can enquire about the status of your transaction. Note that only static URL structures are supported.

Method: POST URL Structure: https://foo.bar/path .

//-> To enable CORS on your server. Follow the following link : https://developers.eko.in/docs/enable-cors