Partner API Specifications - E-KYC Enhancement

Version 2.2, updated 12-November-2021

This partner API specification document is to support customer identification on the selected alternative channels for the purpose of KYC.

Version history

Date	Version	Description	Author
2021-11-12	2.2	• Add the 2 new endpoints below to support DOPA checking screen flow. (section 6 and 7) • Add page break and resize font to make content fix in the page	Piyapon M.
2021-09-15	2.1	• Update API specification to support PDPA consents, face comparison, AML watchlist verification and channel application's icon display	Piyapon M.

Reviewed by Adisorn Chockaumnuai, Phutsita Aphilertrungchut, Anupat Kuekijpaiboon and Nitha Khunwong

1. API list

There are four interfaces available for partner to call as listed below.

API name	Description	HTTP method
OAuth access token	To get access token from Krungsri for API authorization	POST
QR validation for partner	To validate QR or barcode reference together with citizen ID	POST
Submit transaction data	To submit customer information from partner to Krungsri	POST
Transaction reference failure	To update transaction reference as failure status from partner to Krungsri	PUT
Get consent contents	To get PDPA consent contents	GET
Check DOPA	To verify customer information from ID card with DOPA API from Department of Provincial Administration	POST

2. OAuth access token API

APIs are secured with OAuth 2 security standards. It authenticates the users, followed by the user authorizing the API call. This API is to obtain an access token and use it to access protected resources.

`POST /auth/oauth/v2/token`

UAT: https://uat.api.krungsri.com/auth/oauth/v2/token

Request header

Name	Type	Required	Description
Content-Type	String	Yes	application/x-www-form-urlencoded

Request body

Name	Type	Required	Description
scope	String	Yes	Application scopes, defaults to "ndid:alternative"
grant_type	String	Yes	Client credentials grant type
client_id	String	Yes	Client ID
client_secret	String	Yes	Client secret key

Response

Name	Type	Description
access_token	String	Access token
token_type	String	Token type
expires_in	Number	Token expiry time in seconds
scope	String	Application scope

Python sample request

```
def get_oauth_token():
    url = 'https://sandbox.api.krungsri.com/auth/oauth/v2/token'
    headers = {'Content-Type':'application/x-www-form-urlencoded'}
    payload = {
        'grant_type':'client_credentials',
        'scope':'ndid:alternative',
        'client_id':'1704aa7c1063bb4cd3b73...',
        'client_secret':'52ec073v89mk0n...'
    }
    access_token_response = requests.post(url, headers=headers,
       data=payload, verify=False, allow_redirects=False)
    tokens = json.loads(access_token_response.text)
    return tokens['access_token']
```

JSON sample response


```
{
  "access_token": "11da2e69-1774-4e9a-b0ab-103b4669110d",
  "token_type": "Bearer",
  "expires_in": 900,
  "scope": "data:submission"
}
```

3. QR validation for partner API

This service is to request DiiS to validate if such transaction reference still valid. Status code will be returned if it is active, expired or used.

Rule	Business logic	Remark
Validity check	Check if such reference in transaction reference data is valid	Returns "0000" if success "1100" in case of error
Expiry check	Check if such reference is expired where (qrRequestDateTime + qrValidity) > Now	Returns "1101" if it is expired
Rule	**Business logic**	**Remark**
Invalid reference check	Check if such reference matches with identifier registered in DiiS	Returns "1102" if Doesn't match "1100" in case of error

`POST /native/ndid/alternativeSource/qr/validation`

UAT: https://uat.api.krungsri.com/native/ndid/alternativeSource/qr/validation

Request header

Name	Type	Required	Description
Content-Type	String	Yes	application/json; charset=UTF-8
X-Client-Transaction-ID	String	Yes	UUID generated by the client to identify its own request (e.g. a01fa0ae-190a-49d5-9cc1-465f9f9ab0dc)
X-Client-Transaction-DateTime	String	Yes	Date and time on the client side in ISO8601 format (e.g. 2019-12-11T101800.449+0700)
Authorization	String	Yes	Bearer authorization type (OAuth token)
API-Key	String	Yes	Assigned API key for client authentication

Request body

Name	Type	Required	Description
data	BASE64 String	Yes	Encrypted data of identifier and namespace, see section 5 for details
crc	BASE64 String	Yes	Cyclic redundancy check, see section 6 for details

Such encrypted data in the payload must contain the following in JSON data.

Name	Type	Required	Description
reference	String (55)	Yes	Transaction reference
identifier	String (13)	Yes	Citizen ID 13 digits
namespace	String (10)	Yes	Identifier type, defaults to "citizen_id"

Response

Name	Type	Description
statusCode	String (4)	Status code
statusMessage	String (1024)	Status message
data	Object	Data object
data.reference	String (55)	UUID reference number
data.ref1	String (40)	To be printed on slip line 15th
data.ref2	String (40)	To be printed on slip line 16th
data.ref3	String (40)	To be printed on slip line 17th
data.faceCompare	String (1)	The flag informing that customer face photo is required in submitting data (eg. “Y” means Yes - required, “N” means No – not required)
data.channelCode	String(15)	Channel application code for application’s icon reference
data.consents	Array	Array of consent objects containing consent’s content for display if needed [{ type: String, version: String, contentTh: String, contentEn: String }]

JSON sample data before encryption

```
{
    "reference": "XXXXXXXXXXXX",
    "identifier": "0123456789012",
    "namespace": "citizen_id"
}
```

JSON sample success response

```
{
  "statusCode": "0000",
  "statusMessage": "Success",
  "data": {
    "reference": "XXXXXXXXXXXX",
    "ref1": "channel reference number",
    "ref2": "customer reference number",
    "ref3": "transaction type"
    "faceCompare": "Y",
    "channelCode": "KMA"
    "consents": [
        { "type": "103", "version": "2.00.00",
          "contentTh": "<text>", "contentEn": "<text>" }
    ]
  }
}
```

JSON sample error response

```
{
  "statusCode": "1101",
  "statusMessage": "QR code has expired",
  "data": {}
}
```

4. Submit transaction data API

This service is to submit customer information back to Krungsri after completion of the customer activities on partner channel.

`POST /native/ndid/alternativeSource/submit`

UAT: https://uat.api.krungsri.com/native/ndid/alternativeSource/submit

Request header

Name	Type	Required	Description
Content-Type	String	Yes	application/json; charset=UTF-8
X-Client-Transaction-ID	String	Yes	UUID generated by the client to identify its own request (e.g. a01fa0ae-190a-49d5-9cc1-465f9f9ab0dc)
X-Client-Transaction-DateTime	String	Yes	ate and time on the client side in ISO8601 format (e.g. 201912-11T101800.449+0700)
Authorization	String	Yes	Bearer authorisation type (OAuth token)
API-Key	String	Yes	Assigned API key for client authentication

Request body

Name	Type	Required	Description
data	BASE64 String	Yes	Encrypted customer information, see section 5 for details
crc	BASE64 String	Yes	Cyclic redundency check, see section 6 for details

Customer data in the payload should contain the following before encryption.

Name	Type	Required	Description
transactionRef	String (55)	Yes	Channel transaction reference either qrReference or barcodeReference
transactionDateTime	String (14)	Yes	Transaction date/time in YYYYMMDDHHMMSS format
chipNo	String (20)	Yes	Chip number of the citizen ID card
bp1No	String (20)	Yes	Citizen ID card request number
branchId	String (20)	No	Channel branch ID
terminalId	String (20)	No	Channel terminal ID

customerConsents	Array	Yes*	*This field is required when data.faceCompare='Y' returned from QR validation API. Array of customer consents [{ type: string, version: string, consent: string of "Y" - Yes or "N" - No }]
customerImage	BASE64 String	Yes	Customer image in BASE64 string
customerSelfieImage	BASE64 String	Yes*	This field is required when data.faceCompare='Y' returned from QR validation API. Realtime customer face photo. (Recommendation: photo size < 100K, resolution is 680 x 480)
customerSelfieImageFormat	String (10)	Yes*	*This field is required when data.faceCompare='Y' returned from QR validation API. (eg. "JPG","BMP","PNG","JPEG")
customerData	Object	Yes	Customer data object
customerData.mobile	String (10)	No	Mobile phone number
customerData.citizenId	String (13)	Yes	Citizen ID
customerData.prefixTH	String (100)	Yes	Customer prefix name in Thai
customerData.firstNameTH	String (100)	Yes	Customer first name in Thai
customerData.lastNameTH	String (100)	Yes	Customer last name in Thai
customerData.prefixEN	String (100)	Yes	Customer prefix name in English
customerData.firstNameEN	String (100)	Yes	Customer first name in English
customerData.lastNameEN	String (100)	Yes	Customer last name in English
customerData.dateOfBirth	String (8)	Yes	Customer date of birth in YYYYMMDD format
customerData.religion	String (20)	No	Customer religion
customerData.citizenIdCardIssueDate	String (8)	No	Card issued date in YYYYMMDD format
customerData.citizenIdCardExpireDate	String (8)	No	Card expiry date in YYYYMMDD format
customerData.fullAddress	String (200)	No	Customer address on card

customerData.CountryCode	String (3)	No	Country code
customerData.zipCode	String (5)	Yes	Address zip code
customerData.province	String (100)	Yes	Province
customerData.district	String (100)	Yes	District
customerData.subDistrict	String (100)	Yes	Sub district
customerData.road	String (100)	No	Road
customerData.soi	String (50)	No	Soi
customerData.moo	String (2)	No	Moo
customerData.addrNo	String (20)	No	Address no.

Response

Name	**Type**	**Description**
statusCode	String (4)	Status code
statusMessage	String (1024)	Status message
data	Object	Data object
data.reference	String (55)	UUID reference number

JSON sample customer data before encryption

```
 {
   "transactionRef": "BAYMK191200001",
   "transactionDateTime": "20191203203040",
   "chipNo": "6300c10e142f28a1",
   "bp1No": "10994078500",
   "customerImage": "3DEtm3btm0fjti2bdu27cMR27g==",
   "branchId": "",
   "terminalId": "",
   "customerData": {
       "mobile": "0819999999",
       "citizenId": "0123456789012",
       "prefixTH": "นาย",
       "firstNameTH": "ทดสอบชื่อ",
       "lastNameTH": "ทดสอบนามสกุล",
       "prefixEN": "Mr.",
       "firstNameEN": "firstNameTest",
       "lastNameEN": "lastNameTest",
       "dateOfBirth": "19810501",
       "religion": "",
       "citizenIdCardIssueDate": "20200101",
       "citizenIdCardExpireDate": "20220101",
       "fullAddress": "fullAddress",
       "countryCode": "TH",
       "zipCode": "10130",
       "province": "สมุทรปราการ",
       "district": "พระประแดง",
       "subDistrict": "สำโรง",
       "road": "road",
       "soi": "soi",
       "moo": "",
       "addrNo": "1222"
   },
   "customerConsents": [
    {"type": "103", "version": "2.00.00", "consent": "Y"},
    {"type": "105", "version": "2.00.00", "consent": "Y"}
   ],
   "customerSelfieImage": "3DEtmddl+=0fjkldjMr23fg=gRTT",
   "customerSelfieImageFormat": "JPG"
 }
```

JSON sample response

```
{
  "data": {
     "transactionRef": "XXXXXXXXXXXX"
  },
  "statusCode": "0000",
  "statusMessage": "Success"
}
```

5. Transaction reference failure API

This service is to submit the failure result of customer verification occured on the partner side to Krungsri, e.g. customer is blacklisted.

`PUT /native/ndid/alternativeSource/reference/reference/status/failure`

UAT: https://uat.api.krungsri.com/native/ndid/alternativeSource/reference/status/failure

Request header

Name	Type	Required	Description
Content-Type	String	Yes	application/json; charset=UTF-8
X-Client-Transaction-ID	String	Yes	UUID generated by the client to identify its own request (e.g. a01fa0ae-190a-49d5-9cc1-465f9f9ab0dc)
X-Client-Transaction-DateTime	String	Yes	ate and time on the client side in ISO8601 format (e.g. 201912-11T101800.449+0700)
Authorization	String	Yes	Bearer authorisation type (OAuth token)
API-Key	String	Yes	Assigned API key for client authentication

Request body

Name	Type	Required	Description
reference	String	Yes	Transaction reference
data	BASE64 String	Yes	Encrypted customer information, see section 5 for details
crc	BASE64 String	Yes	Cyclic redundency check, see section 6 for details

Data in the payload should contain the following before encryption.

Name	Type	Required	Description
identifier	String (13)	Yes	Citizen ID 13 digits
namespace	String (10)	Yes	Identifier type, defaults to "citizen_id"
detailError	Object	Yes	Detail error data object
detailError.code	No	Error code from partner backend	
detailError.message	No	Error code from partner backend	

Response

Name	Type	Description
result	Object	status object
statusCode	String (4)	Status code
statusMessage	String	Status message
data	Object	Data object

JSON sample data before encryption

```
{
    "identifier": "0123456789012",
    "namespace": "citizen_id",
    "detailError": {
        "code": "01",
        "message": ""
    }
}
```

JSON sample response

```
{
    "data": {
        "reference":
"lFOL5cCwcPvQmySXbMcsURl3cKLkje9NaedV53I79adlSpScK5wD7aa"
    },
    "statusCode": "0000",
    "statusMessage": "Success"
}
```

6. Get consent contents API

This service is to send customer information from ID card to validate with DOPA API.

`GET /native/ndid/alternativeSource/consents/{ref}`

UAT: https://uat.api.krungsri.com/native/ndid/alternativeSource/consents/{ref}

Currently the available value of {ref} in url is 'dopa'.

Request header

Name	Type	Required	Description
Content-Type	String	Yes	application/json; charset=UTF-8
X-Client-Transaction-ID	String	Yes	UUID generated by the client to identify its own request (e.g. a01fa0ae-190a-49d5-9cc1-465f9f9ab0dc)
X-Client-Transaction-DateTime	String	Yes	ate and time on the client side in ISO8601 format (e.g. 201912-11T101800.449+0700)
Authorization	String	Yes	Bearer authorization type (OAuth token)
API-Key	String	Yes	Assigned API key for client authentication

Response

Name	Type	Description
statusCode	String (4)	Status code
statusMessage	String	Status message
data	Object	Data object
data.ref	String(25)	Consent reference
data.consents	Array	Array of consent objects containing consent's content for display if needed [{ type: String, version: String, contentTh: String, contentEn: String }]

JSON sample response

```
{
   "statusCode": "0000",
   "statusMessage": "success",
   "data": {
      "ref": "dopa"
      "consents": [
           { "type": "103", "version": "2.00.00",
             "contentTh": "<text>", "contentEn": "<text>" }
        ]
   }
}
```

7. Check DOPA API

This service is to send customer information from ID card to validate with DOPA API.

`POST /native/ndid/alternativeSource/dopa`

UAT: https://uat.api.krungsri.com/native/ndid/alternativeSource/dopa

Request header

Name	Type	Required	Description
Content-Type	String	Yes	application/json; charset=UTF-8
X-Client-Transaction-ID	String	Yes	UUID generated by the client to identify its own request (e.g. a01fa0ae-190a-49d5-9cc1-465f9f9ab0dc)
X-Client-Transaction-DateTime	String	Yes	ate and time on the client side in ISO8601 format (e.g. 201912-11T101800.449+0700)
Authorization	String	Yes	Bearer authorization type (OAuth token)
API-Key	String	Yes	Assigned API key for client authentication

Request body

Name	Type	Required	Description
data	BASE64 String	Yes	Encrypted customer information, see section 5 for details
crc	BASE64 String	Yes	Cyclic redundancy check, see section 6 for details

Customer data in the payload should contain the following before encryption.

Name	Type	Required	Description
customerConsents	Array	Yes	Array of customer consents for check DOPA [{ type: string, version: string, consent: string of “Y” - Yes or “N” - No }]
chipNo	String (20)	Yes	Chip number of the citizen ID card
bp1No	String (20)	Yes	Citizen ID card request number
branchId	String (20)	No	Channel branch ID
terminalId	String (20)	No	Channel terminal ID
customerImage	BASE64 String	Yes	Customer image in BASE64 string

customerData	Object	Yes	Customer data object
customerData.mobile	String (10)	No	Mobile phone number
customerData.citizenId	String (13)	Yes	Citizen ID
customerData.prefixTH	String (100)	Yes	Customer prefix name in Thai
customerData.firstNameTH	String (100)	Yes	Customer first name in Thai
customerData.lastNameTH	String (100)	Yes	Customer last name in Thai
customerData.prefixEN	String (100)	Yes	Customer prefix name in English
customerData.firstNameEN	String (100)	Yes	Customer first name in English
customerData.lastNameEN	String (100)	Yes	Customer last name in English
customerData.dateOfBirth	String (8)	Yes	Customer date of birth in YYYYMMDD format
customerData.religion	String (20)	No	Customer religion
customerData.citizenIdCardIssueDate	String (8)	No	Card issued date in YYYYMMDD format
customerData.citizenIdCardExpireDate	String (8)	No	Card expiry date in YYYYMMDD format
customerData.fullAddress	String (200)	No	Customer address on card
customerData.CountryCode	String (3)	No	Country code
customerData.zipCode	String (5)	Yes	Address zip code
customerData.province	String (100)	Yes	Province
customerData.district	String (100)	Yes	District
customerData.subDistrict	String (100)	Yes	Sub district
customerData.road	String (100)	No	Road
customerData.soi	String (50)	No	Soi
customerData.moo	String (2)	No	Moo
customerData.addrNo	String (20)	No	Address no.

Response

Name	Type	Description
statusCode	String (4)	Status code
statusMessage	String (1024)	Status message
data	Object	Data object
data.transactionRef	String (55)	Transaction reference number

JSON sample customer data before encryption


```
{
  "chipNo": "6300c10e142f28a1",
  "bp1No": "10994078500",
  "customerImage": "3DEtm3btm0fjti2bdu27cMR27g==",
  "branchId": "",
  "terminalId": "",
  "customerData": {
      "mobile": "0819999999",
      "citizenId": "0123456789012",
      "prefixTH": "นาย",
      "firstNameTH": "ทดสอบชื่อ",
      "lastNameTH": "ทดสอบนามสกุล"
      "prefixEN": "Mr.",
      "firstNameEN": "firstNameTest",
      "lastNameEN": "lastNameTest",
      "dateOfBirth": "19810501",
      "religion": "",
      "citizenIdCardIssueDate": "20200101",
      "citizenIdCardExpireDate": "20220101",
      "fullAddress": "fullAddress",
      "countryCode": "TH",
      "zipCode": "10130",
      "province": "สมุทรปราการ",
      "district": "พระประแดง",
      "subDistrict": "สำโรง",
      "road": "road",
      "soi": "soi",
      "moo": "",
      "addrNo": "1115",
  },
  "customerConsents": [ {"type": "105", "version": "2.00.00", "consent": "Y"}]
}
```

JSON sample response


```
{
  "data": {
     "transactionRef": "XXXXXXXXXXXX"
  },
  "statusCode": "0000",
  "statusMessage": "Success"
}
```

8. Payload encryption

By sending customer information through Krungsri Open API, values in request payload must be encrypted with "**AES/CBC/PKCS7Padding**" algorithm having AES key size 256 bits and IV 128 bits with Base64 encoded. Then put on the data field in the request body.

Java sample code

```
import javax.crypto.Cipher;
import javax.crypto.Mac;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import java.security.MessageDigest; import
java.security.SecureRandom;
import java.util.Base64;

public class AESMain {
   private  static  final  String  key  =  "32KeyValue.Secret1234567890asdfg";
   private static final String initVector = "16IVValue.Secret";

   public static String encrypt(String value) {
      try {
           IvParameterSpec iv = new IvParameterSpec(initVector.getBytes(
"UTF-8"));
           SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(
"UTF-8"), "AES");

           Cipher cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
           cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);

           byte[] encrypted = cipher.doFinal(value.getBytes());
           return Base64.getEncoder().encodeToString(encrypted);
      } catch (Exception ex) {
           ex.printStackTrace();
      }
      return null;
  }

  public static String decrypt(String encrypted) {
     try {
          IvParameterSpec iv = new IvParameterSpec(initVector.getBytes(
"UTF-8"));
          SecretKeySpec skeySpec = new SecretKeySpec(key.getBytes(
"UTF-8"), "AES");

          Cipher cipher = Cipher.getInstance("AES/CBC/PKCS7Padding");
          cipher.init(Cipher.DECRYPT_MODE, skeySpec, iv);
          byte[] original =
cipher.doFinal(Base64.getDecoder().decode(encrypted));
          return new String(original);
     } catch (Exception ex) {
          ex.printStackTrace();
     }
     return null;
   }
}
```

Python sample code

```
import json
from base64 import b64encode, b64decode
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad


def do_encrypt(data,key,init_vector):
    try:
        cipher = AES.new(key, AES.MODE_CBC, init_vector)
        ct_bytes = cipher.encrypt(pad(data, 16, style='pkcs7'))
        cipher_text = b64encode(ct_bytes).decode('utf-8')
        return json.dumps({'cipherText':cipher_text})
    except ValueError or KeyError:
        logging.exception('Encryption fails!')

def do_decrypt(data,key,init_vector):
    try:
        data_json = json.loads(json.dumps(data))
        cipher_text = b64decode(data_json['cipherText'])
        cipher = AES.new(key, AES.MODE_CBC, init_vector.encode('utf-8'))
        return unpad(cipher.decrypt(cipher_text), 16)
    except ValueError or KeyError:
        logging.exception('Decryption fails!')

KEY = "32KeyValue.Secret1234567890asdfg"
INIT_VECTOR = "16IVValue.Secret"
customer_data_bytes = json.dumps(customer_data).encode('utf-8')
encrypted_customer_data = do_encrypt(customer_data_bytes, KEY.encode('utf8'),
INIT_VECTOR.encode('utf-8'))
```

9. Cyclic redundancy check (CRC)

A checksum algorithm is used to detect inconsistency of the submitted data. Partner to generate an AES key while public key will be provided by Krungsri for RSA encryption (RSA/ECB/OAEPWithSHA-1AndMGF1Padding).

Java sample code

```
Import javax.crypto.Cipher;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.security.*;
import java.security.spec.X509EncodedKeySpec;
import java.util.Base64;
import java.util.List;
import java.util.stream.Collectors;
import javax.crypto.IllegalBlockSizeException;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.BadPaddingException;
import java.security.spec.InvalidKeySpecException; import java.io.IOException;


public class RSAMain {

   public static byte[] doEncrypt(String data2Encrypt) throws
        IOException, InvalidKeySpecException, BadPaddingException,
        IllegalBlockSizeException, InvalidKeyException, NoSuchPaddingException,
        NoSuchAlgorithmException {
           List<String> publicKeyContent = Files.readAllLines(
                                                Paths.get("devalt.pub.pem"));

           String publicKeyInPEM = publicKeyContent.stream()
               .filter(
                  line -> !line.startsWith("---")).collect(Collectors.joining()
               );

           byte[] keyBytes = Base64.getDecoder().decode(publicKeyInPEM);
           X509EncodedKeySpec spec = new X509EncodedKeySpec(keyBytes);
           KeyFactory kf = KeyFactory.getInstance("RSA");
           PublicKey pubKey = kf.generatePublic(spec);
           Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWithSHA-1AndMGF1Padding");
           cipher.init(Cipher.ENCRYPT_MODE, pubKey);

        return Base64.getEncoder().encode(cipher.doFinal(data2Encrypt.getBytes()));
   }

   public static void main(String[] args) throws Exception {
           String secretKey = "32KeyValue.Secret1234567890asdfg@16IVValue.Secret";
           byte[] cipherText = doEncrypt(secretKey);
           System.out.println(new String(cipherText));
       }
   }
```

10. Generic error codes

JSON sample error response

```
{
  "error": {
    "code": "100",
    "message" : "Description and detail of error",
    "messageTH" : "ข้อความที่ต้องการให้แสดงต่อผู้ใช้งานเป็นภาษาไทย                    ",
    "serverDateTime": "2019-09-01T18:39:35.45+07:00",
    "clientTransactionID": "UUID submitted by the Client",
    "serverTransactionID": "UUID generated by the Server"

  }
}
```

Note that serverDateTime, clientTransactionID and serverTransactionID are equal to the respective HTTP headers X-Server-DateTime, X-Client-Transaction-ID and X-Server-Transaction-ID.

The code value and its associated description gives a clear indication of the problem. The table below describes the standard possible problems and how to address them. It has been designed with the minimum HTTP status to handle.

HTTP status	Reason phrase code	Details
200	OK	Standard response for successful HTTP requests
400	Bad Request	- Invalid request structure (the request cannot be parsed) - Invalid parameter in the request (i.e. account number cannot be found) - Submitted data does not match the specification, it can also be considered as a possible attack
401	Unauthorized	Authentication failed
500	Internal Server Error	Transaction failed. Something went wrong technically in the gateway or in the back-end, or simply some business logic prevents execution (i.e. fund insufficient)
503	Service Unavailable	Krungsri server side is currently saturated

API GW status code table

API error code	**Message**

-3	Invalid Content-Type header
-4	Schema validation failed
-6	X-Client-Transaction-ID does not match UUID format
-7	Invalid Accept header
-19	Threat detected (invalid signature)
-20	Threat detected (code injection)
-21	Threat detected (request empty, overall size or field length exceed maximum size)
-22	Threat detected (message size too large)
-23	Threat detected (message replay detected)
-24	Service unavailable
-25	Threat detected (Protect against message replay: Invalid Request. X-Client-Transaction-ID, XClient-DateTime and X-Signature are required)
-26	Threat detected (Protect against message replay: Request X-Client-DateTime has exceeded ${offset} seconds limit with server time)
-27	Threat detected (SQL attack)
-28	Duplicated name in the JSON request
-100	Invalid token (unknown, expired, scope not granted, etc)
-101	Invalid session state for the token
-102	API Key invalid or not authorised to call this API
-103	Incoming source IP does not match your identity
-104	Service requires basic authentication
-105	Authentication failed
-106	API-Key and OAuth token come from a different application
-107	API-Key header is missing or is empty
-108	Account plan quota exhausted
-200	Internal server error
-201	Connectivity problem between API GW and EAI including timeout
-202	Backend problem, an invalid response has been returned from backend

-300	Current load of this API is too high. Server cannot process the request right now. Please retry later
-301	Current load of this API back-end is too high. Server cannot process the request right now. Please retry later

DiiS response code table

Type	Response code	Description
Success	0000	Success response
QR module	1100	Invalid QR code
	1101	QR code has expired
	1102	Invalid reference
	1103	QR code is cancelled
Submit transaction module	1200	Invalid transaction reference
	1201	DOPA service fails
	1202	DOPA information is invalid
	1301	Callback notification to channel failed
	1302	Customer info does not match with the registered info on channel application
	1303	Callback notification to KMA failed
	1399	Cannot connect channel's callback
	1402	Data Error (AES decryption failed) **the corrected data can be resubmitted before the expired date*
	1501	Data Error (RSA decryption failed) **the corrected data can be resubmitted before the expired date*
	1600	Transaction reference not found
	1601	QR code is cancelled

	1602	Such transaction reference has been used and a message "Transaction reference already used" returns
	1700	Blacklist service failed
	1799	Cannot connect to Blacklist service **please retry summitting the data again later*
	1899	Cannot connect to Face Recognition service **please retry summitting the data again later*
	1900	Customer consents needed
Invalid Identifier	2001	Citizen Id mismatch **the corrected data can be resubmitted before the expired date*
	2002	Biometric comparison failed
	2003	Customer is in blacklist
Invalid biometric data	9300	Biometric selfie data is required **the corrected data can be resubmitted until maximum limit is reached*
(Note: Maximum attempts of resubmitting new data is limited to 5)	9301	Biometric comparison failed. **the corrected data can be resubmitted until maximum limit is reached*
	9302	Biometric comparison error **the corrected data can be resubmitted until maximum limit is reached*
Unauthorized request	9400	Data Error **the corrected data can be resubmitted before the expired date*
	9401	Unauthorized (Invalid API key)
	9402	Unauthorized trusted source
	9403	Unauthorized channel or product mismatch
General error	9997	Time out, cannot contact backend system
	9998	Invalid schema
	9999	Cannot contact backend **please retry summitting the data again later*

11. Glossary

Term	Description
API	Application programming interface
API key	Code passed in by computer programs calling an application programming interface (API) to identify the calling program, its developer, or its user to the Web site. API keys are used to track and control how the API is being used, for example to prevent malicious use or abuse of the API.
DiiS	Digital ID integration system, it is system that provide services for generate QR and receive the customer data from partner application
RSA public key	One of the first public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public, and it is different from the decryption key which is kept secret (private)