Security & Encryption

Client Authentication

The API uses **OAuth2.0** as a security measure to authenticate requests.

Transport Encryption

The connection between the client applications and the API is secured with TLS/SSL.

It is recommended that the URL domain is compatible for both testing and production to ensure that during the testing stage, notification configuration meets FPX requirements.

INFO

Our APIs only support **TLS 1.2**

Message Signature Generation

INFO

You can also find our message signature SDK or sample apps in **resources** section.

Transaction will be signed using an asymmetric (private key) cryptopraphy mechanism. Messages are signed based on key fields rather than the whole message. The signature generated is then inputted into the message `X-Signature` field in the message header. The key fields will be based on each of the different message formats as shown below.

PKI sign algorithm, signature method RSA-SHA 1 Public Key Signature Algorithm is used to compute a Signature value.

All message formats will require to include both the header and body fields to be built into the signature. The steps will be as below :

Step 1: Construct the source string

Based on the request message, respective fields value will be appended together following the order specified in below section.

Step 2: Sign the source string

Sign the constructed source string with participant's private key.

Take the signed value and populate it into the `X-Signature` header field.

Java PHP

```java
public class FPXSignature {

  public static String signDataUsingInternalKey(
    String pvtKeyFileName,
    String dataToSign,
    String signatureAlg
  )
    throws IOException, NoSuchAlgorithmException, NoSuchProviderException,
InvalidKeyException, SignatureException {
    PrivateKey privateKey = getPrivateKey2(pvtKeyFileName);
    Signature signature = Signature.getInstance(signatureAlg, "BC");
    signature.initSign(privateKey);
    signature.update(dataToSign.getBytes());
    byte[] signatureBytes = signature.sign();

    return byteArrayToHexString(signatureBytes);
  }

  public static String byteArrayToHexString(byte b[]) {
    StringBuffer sb = new StringBuffer(b.length * 2);
    for (int i = 0; i < b.length; i++) {
      sb.append(hexChar[(b[i] & 0xf0) >>> 4]);
      sb.append(hexChar[b[i] & 0x0f]);
    }
    return sb.toString();
  }
}
```

Message Signature Fields

Bank Webview Services

DCL Inquiry

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant

CCL Inquiry

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant

DCL Authorize

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant
`transactionId`	URI parameter	Unique ID generated by FPX for each payment request received from Merchant
`action`	URI parameter	Action code. Different actions could be initiated according to the transaction status

CCL Authorize

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant
`transactionId`	URI parameter	Unique ID generated by FPX for each payment request received from Merchant
`action`	URI parameter	Action code. Different actions could be initiated according to the transaction status

Downtime Inquiry

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant
`model`	URI parameter	Business model

Initiate Downtime

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant
`model`	URI parameter	Business model
`periodType`	Body Parameter	Downtime period type
`startDate`	Body Parameter	Downtime start date
`startTime`	Body Parameter	Downtime start time
`endDate`	Body Parameter	Downtime end date
`endTime`	Body Parameter	Downtime end time

Downtime Cancel

Request

Field	Type	Description
`bankId`	URI parameter	Bank code of the Participant
`schedulerId`	URI parameter	Unique identification number assigned to the downtime request

FPX Services

Differ from Bank Webview Services, this service requires all fields to be signed before sending to the FPX under `fpx_checkSum` field.

Given below are the steps to sign a transaction request to FPX Services:

Step 1 – Construct the source string

a. The source string should be formed with **all data element** values.

b. The values should then be sorted by their data element name, in **ascending** order.

c. Each element value should be **separated by a “I” (pipe) character** in between them.

Below is format of source string, in ascending order:

fpx_msgTokenIfpx_msgTypeIfpx_sellerExIdIfpx_version

Below is sample of source string, constructed from the sample message

01IBEIEX0002200I7.0

Step 2 – Sign the source string

a. Sign the constructed source string with Merchant private key.

INFO

This steps applicable to all FPX Services APIs.