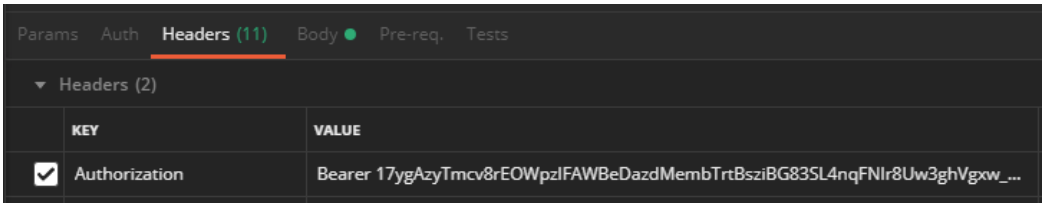
Request Signature

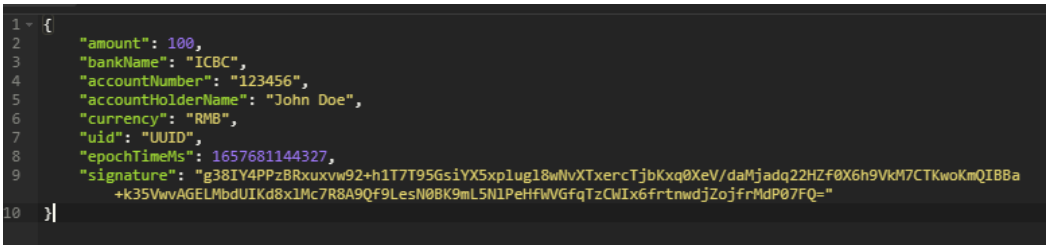
1. 3 keys will be used

API key	Pass into all http request header “Authorization” value “Bearer [api key]”
SIGN key	Key used to hash signature
PUBLIC key	Public key to encrypt signature

2. Signing request
    a. All request simple value will be included into signature (string, number, boolean excluding objects and arrays)
    b. Sort parameter keys in alphabetical order and concatenate with the format key1=val1&key2=val2
    c. Hash the concatenated string with HmacSHA256 using sign key generated above
    d. Encrypt the signature using public key generated above
    e. Only parameter type String, Integer, Double & Long will be used in the signature
3. Sample
signature = HmacSHA256(“accountHolderName=John Doe&accountNumber=123456&amount=100&bankName=ICBC&currency=RMB&epochTimeMs=1657681144327&uid=UUID”) sign with SIGN key & encrypt with PUBLIC key
    a. Adding API key to Authenticate the request



    b. Request body



    c. Example signature method in java

```
@Override
public String sign(MFacto4JApiRequestDto requestDto) throws Exception {
    TreeMap<String, Object> payload = new TreeMap<>(OM.convertValue(requestDto, Map.class));
    String sign = this.sign(payload);
    return this.encryptResponseSignature(sign);
}
```

```java
private String sign(TreeMap<String, Object> payload) {
    if(!payload.containsKey(MFacto4JConstants.TRANSACTION_TIME_PARAM_NAME))
        payload.put(MFacto4JConstants.TRANSACTION_TIME_PARAM_NAME, Instant.now().toEpochMilli());
    List<String> payloadList = new ArrayList<>(payload.size());
    payload.forEach((key, val) -> {
        // only include primitive or primitive wrapper variables to signature, exclude arrays
        if(val != null && isSimpleValue(val.getClass())) payloadList.add(StringUtils.joinWith( delimiter: "=", key, val));
    });
    if (payloadList.isEmpty()) throw new RuntimeException("Payload list empty");
    String hashString = StringUtils.joinWith( delimiter: "&", payloadList.toArray(new Object[0]));
    return new String(Base64.getEncoder().encode(Hashing.hmacSha256(signKey.getBytes(StandardCharsets.UTF_8))
            .hashString(hashString, StandardCharsets.UTF_8).asBytes()));
}
```

```java
public String encryptResponseSignature(String signature) throws Exception{
    PublicKey thirdPartyPubKey = KeyFactory.getInstance( algorithm: "RSA").generatePublic(
            new X509EncodedKeySpec(Base64.getDecoder().decode(
                    base64PublicKey
            ))
    );
    Cipher cipher = Cipher.getInstance( transformation: "RSA/ECB/PKCS1Padding");
    cipher.init(Cipher.ENCRYPT_MODE, thirdPartyPubKey);
    cipher.update(signature.getBytes());
    byte[] cipherText = cipher.doFinal();
    return new String(Base64.getEncoder().encode(cipherText));
}
```

APIs

1. Create account

API Url : https://?????????/api/maccount/create
Request Method : POST
Request Body :

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
accsaId	Input insert by user to bind it to push notification server
name	Account Name

Request Body Sample :

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "accsaId": "3FA64ce9",
  "name": "string"
}
```

Response :

Parma	Description
statusCode	Status code
status	HTTP status
signature	Signature for 3rd party to validate
epochTimeMs	Current timestamp in millisec
message	Response message
maccountId	Binded account ID

Response Sample (HTTP:200, OK):

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {
    "maccountId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
  }
}
```

Response Sample (HTTP:401, UNAUTHORIZED or HTTP:406, NOT ACCEPTABLE):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {}
}
```

Response Sample (HTTP:500, INTERNAL SERVER ERROR):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```

2. For client to generate QR code.

API Url : https://?????????/api/qr/generate

Request Method: POST

Request Body :

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
callback	Third party callback
params	Additional parameter to include in the QR

Request Body Sample :

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "callback": "string",
  "params": {
    "additionalProp1": "string",
    "additionalProp2": "string
  }
}
```

Response :

Parma	Description
statusCode	Status code
status	HTTP status
message	Response message
signature	Signature for 3rd party to validate
epochTimeMs	Current timestamp in millisec
id	Binded account ID
base64Qr	Base64 qr code

Response Sample (HTTP:200, OK):

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6",
    "base64Qr": "string"
  }
}
```

Response Sample (HTTP:401, UNAUTHORIZED or HTTP:406, NOT ACCEPTABLE):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {}
}
```

Response Sample (HTTP:500, INTERNAL SERVER ERROR):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```

3. This api is used by client to send push notification to the user.

API Url : https://?????????/api/mnotification/push
Request Method : POST
Request Body :

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
title	Title of the pop-up notification
description	Description of pop-up notification
header	Header on the notification page
actionList	List of actions for the notification
detailList	Additional details show on the notification page
maccountId	Binded account ID

Request Body Sample :

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "title": "string",
  "description": "string",
  "header": "string",
  "actionList": [
    {
      "action": "string",
      "url": "string",
      "color": "string"
    },
  "detailList": [
    {
      "key": "string",
      "value": "string"
    }
  ]
  "maccountId": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
```

```
}
```

Response :

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
statusCode	Status code
status	HTTP status
message	Response message
id	Push notification ID

Response Sample (HTTP:200, OK):

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {
    "id": "3fa85f64-5717-4562-b3fc-2c963f66afa6"
  }
}
```

Response Sample (HTTP:401, UNAUTHORIZED or HTTP:406, NOT ACCEPTABLE):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {}
}
```

Response Sample (HTTP:500, INTERNAL SERVER ERROR):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```

4. This api is used to upload the mclient's icon.

API Url : https://?????????/api/client/icon/upload

Request Method : POST

Request Body :

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
file	Icon image. Only support **.PNG** with size less than **1MB**.

Request Body Sample :

```
Unset
file: binary,
request: {
  "epochTimeMs": 0,
  "signature": "string"
}
```

Response :

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
statusCode	Status code
status	HTTP status
message	Response message
result	Return success if successfully uploaded

Response Sample (HTTP:200, OK):

```
Unset
{
```

```
  "epochTimeMs": 0,
  "signature": "string",
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```


Response Sample (HTTP:401, UNAUTHORIZED or HTTP:406, NOT ACCEPTABLE):


```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {}
}
```


Response Sample (HTTP:500, INTERNAL SERVER ERROR):


```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```

5. This api is used by client to test client's callback url.

API Url：https://?????????/api/client/checkCallback

Request Method: POST

Request Body：

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
callback	Callback from client

Request Body Sample：

```
Unset
{
  "epochTimeMs": 0,
  "signature": "string",
  "callback": "string"
}
```

Response：

Parma	Description
epochTimeMs	Current timestamp in millisec
signature	Signed key using sign key encrypted with public key
statusCode	Status code
status	HTTP status
message	Response message
result	Return success once submit a callback event.

Response Sample (HTTP:200, OK):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```

Response Sample (HTTP:401, UNAUTHORIZED or HTTP:406, NOT ACCEPTABLE):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {}
}
```

Response Sample (HTTP:500, INTERNAL SERVER ERROR):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": "string"
}
```

6. Get current system time

API Url：https://?????????/api/client/time

Request Method: GET

Response：

Parma	Description
statusCode	Status code
status	HTTP status
message	Response message
result	Return current system time

Response Sample (HTTP:200, OK):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": 0
}
```

Response Sample (HTTP:401, UNAUTHORIZED or HTTP:406, NOT ACCEPTABLE):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
  "result": {}
}
```

Response Sample (HTTP:500, INTERNAL SERVER ERROR):

```
Unset
{
  "statusCode": 0,
  "status": "string",
  "message": "string",
```

```
  "result": "string"
}
```