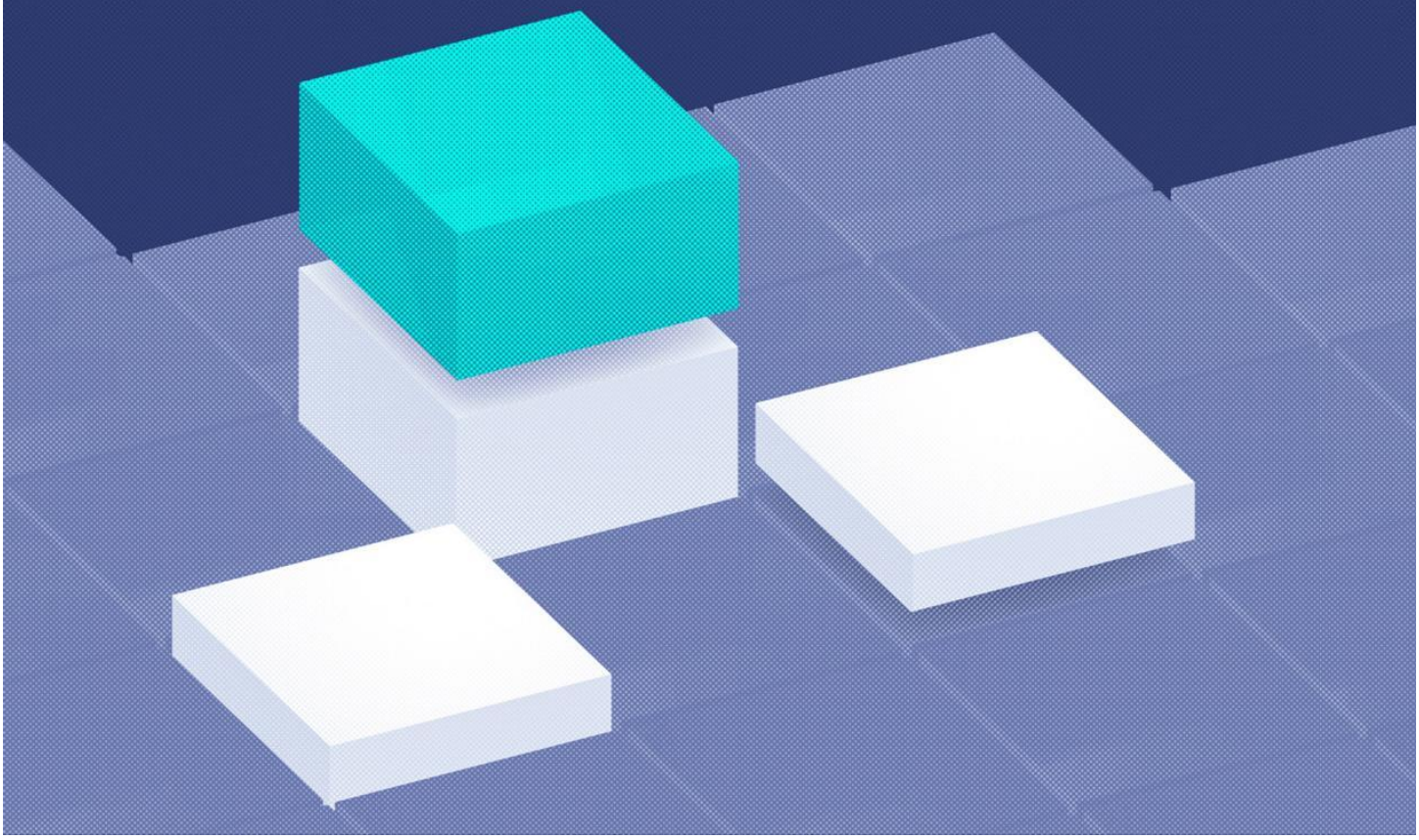
WalletsNet.
Instant Payment Network

RESTful API

Version 1.0
Jan 2021



Introduction

WalletsNet® enables digital wallets to be connected to each other in a Standard, Speedy, Simple and Secure way around the world.

This document will show you how to join WalletsNet through the API.

Get started

This chapter is an overview of the integration process of WalletsNet.

step 1. Read WalletsNet® Message Specifications carefully.

step 2. Build a ISO-20022 message.

step 3. Read Digital signature, add signature to message.

step 4. Read Message API section, try send message to WalletsNet until WalletsNet returns success.

step 5. Receive WalletsNet's message and Read WalletsNet® Message Specifications find next step.

step 6. Repeat step 1 to 5 until transaciton successed.

Digital signature

To guarantee that data have not been altered in transmission, digital signature and encryption mechanism can be adopted. For all messages, the digital signature is mandatory. In addition, data encryption is required if sensitive information is enclosed in the message.

The signature algorithm used for data transmission is RSA 256. You can use RSA256 for creating or validating signatures. When creating a signature, calculate a sha256 digest first, and then encrypt the digest by using RSA algorithm. The recommended RSA key size is 2048 bits.

Prepare keys

You can generate a public and private RSA key pair like this:

```
openssl genrsa -des3 -out private.pem 2048
```

That generates a 2048-bit RSA key pair, encrypts them with a password you provide and writes them to a file. You need to next extract the public key file. You will use this, for instance, on your web server to encrypt content so that it can only be read with the private key.

Export the RSA Public Key to a File

```
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
```

You can use less to inspect each of your two files in turn:

```
less private.pem to verify that it starts with a
-----BEGIN RSA PRIVATE KEY-----
less public.pem to verify that it starts with a
-----BEGIN PUBLIC KEY-----
```

Send & Protect keys

Send public.pem to WalletsNet, The public key can be distributed anywhere or embedded in your web application scripts.

Protect private.pem, it’s important to keep the private key backed up and secret.

Create a signature

Complete the following steps to create a signature:

1. Obtain the private key. See Preparing keys for details.

2. Create a IOS-20022 message.

3. Create the string to sign

```
string-to-sign = "message_id=" + {message.head.BizMsgIdr} + "&member_id=" + {message.head.Fr.FIId.FinInstnId.ClrSysMmbId.MmbId} + "&scheme=" + {message.head.MsgDefIdr} + "$created=" + {message.head.CreDt}
```

4. Generate the signature. Use the algorithm and private key obtained in step1 to generate the signature. The following example assumes that RSA256 algorithm is used to generate the signature:

```
signature=base64UrlEncode(sha256withrsa(string-to-sign))
```

5. Add the signature to message's header.

```
message.body.AppHdr.Sgntr.Signature.SignatureValue = signature
```

Message API

Request URL:

Production: http://api.walletsnet.com/message

Sandbox: http://sandbox.walletsnet.com/message

Method: POST

Request parameters

🎬 **scheme**: Enum Required Request scheme. Possible values are:

- **JSON:** Indicates that the message scheme is JSON
- **XML:** Indicates that the message scheme is XML

🎬 **message:** Text Required ISO-20022 Message String

Response parameters

- **resultCode:** Enum Required Result status. Possible values are:
  - **S:** Indicates that the result status is successful.
  - **F:** Indicates that the result status is failed.
  - **U:** Indicates that the result status is unknown.
- **resultMessage:** String(64) Optional Result message

More information

The message API return "Success" only means that WalletsNet received the message correctly, it does not mean that the business was executed successfully.

You will need to refer to the WalletsNet® Message Specifications for the next step.

End of
Document