Anypay Integration Guide

So you want your own digital coins to circulate as money in the retail economy. After much research you discovered that Anypay is the simplest, most fun point of sale app for digital cash. Now you are ready to roll up your sleeves and integrate, so there are a few steps you need to take.

About Anypay Invoices

At checkout in a retail business Anypay generates an invoice with a unique address to identify the customer's payment. The address information is encoded in a URI and corresponding QR code to communicate properly with the customer's wallet. The URI includes the price of the currency denominated in the fiat currency of the business manager's choosing.

For payments in your coin to be effectively communicated and identified your system must be able to generate unique addresses for every single payment. Most crypto currency systems use hierarchically-derived public key addresses, while others use a single account plus some random nonce.

Once a customer scans the invoice and sends payment in the correct amount, Anypay will detect payment. For a correct payment, the system will designate that invoice as paid, and notify the cashier and customer of successful payment. In the case of under-payment or over-payment the customer and cashier will be notified of the disparity.

Coin Integration API

Since you are the master of your coin's logical and functional domain you are responsible for everything related to your coin, and provide the various functions with an authenticated HTTP-based service.

The service you implement will expose several endpoints for initial system-wide setup of your coin, configuring a new business to accept your coin, generating new addresses for each customer payment, and notifying Anypay of new payments for an invoice.

Initial Coin Setup

You will provide a single URL for your coin's integration service, which will respond to the various requests.

In this example we will construct a service that provides Dogecoin (DOGE) at the following URL.

`https://dogeforretail.com/api/anypay-integration`

Coin Info and Logo

The coin has a name and a currency code, as well as a logo displayed to the business manager and customers during normal day-to-day use.

`GET /coininfo`

- logo – url of image file to display as logop
- name – long-form name of coin (ex: Dogecoin)
- code – currency code (DOGE, BCH, etc)

Response:

```
{
  "coin": {
    "logo": "https://dogeforretail.com/coinlogo.png",
    "name": "Dogecoin",
    "code": "DOGE"
  }
}
```

Prices

Anypay simplifies accepting new forms of money by automatically calculating the amount to pay given a price in terms of a more familiar asset, JPY or EUR for example. In order to do that we must understand what the current price of your asset is. Do not worry about maintaining prices in terms of all currencies, your coin may have a single price to a well-known asset in which case we will compute the price based on our existing price table.

`GET /prices`

Response:

```
{
  "prices": {
    "DOGE/EUR": 8177.2,
    "DOGE/JPY": 2134343983.33,
    "DOGE/USD": 6812.33
   }
}
```

Business Public Key Address Validation

For simplicity Anypay avoids holding on to customer funds in any long-term custodial fashion by acting as a router for payments from the customer to the merchant. Business managers provide the payment addresses to which they would like to receive settlements. Then Anypay instantly routes payments from the customer to this payment address.

Upon initial business setup the payment address must be properly validated in order to ensure payments are routed correctly to to the business's own wallet. This address generally remains the same during the lifetime of a given business's operation with Anypay, unless they decide to update it from time to time.

```
GET /addresses/Dg33I3211HHg9F839ym3F399f17yekk
```

Response:

```
{
  "valid" : false
}

or

{
  "valid" : true
}
```

Routing Payments to Unique Addresses

Now to the bulk of the behavior your integration service provides to Anypay, generating unique addresses, detecting payment at that address, forwarding the payment along to the business's wallet, and notifying Anypay of payment success.

Generally there exist two classes of address, normal public keys and extended public keys. Normal public keys require payment forwarding whereas extended public keys enable direct customer-to-merchant payments in a more trustless manner.

When a business manager provides a normal public key address their address remains static. In that case your service must generate unique addresses for each payment and forward payments to that address. Mathematically each invoice address will be unrelated to the merchant's settlement address.

When using extended public keys every unique invoice address is mathematically derived from the merchant's public key they provide at setup. While your system is still responsible for generating a unique address for every invoice, it will not be responsible for forwarding payments since all derived addresses belong to the businesses's wallet directly.

For this reason extended public keys are always preferred when available. For every new invoice Anypay will ask your service for a new unique invoice address by providing the business's settlement address and a unique nonce. The unique nonce will begin at 0 and increment by one for every invoice generated.

Normal Public Keys

`POST /addresses`

Here the destination is the address to which the business will ultimately receive settlement. This address is never displayed to the customers.

Respond with an addresses. The `input` address will be the address to which payment is made by a customer. When payment is received to the input address your system is responsible for relaying the payment on to the destination address.

Response:

```
{
  "address": "Xkp5r7Luq9sGSLbTP2Mss2k3HSkPuATXkx"
}
```

Extended Public Keys

`POST /addresses`

In the case of extended public keys payments go directly from the customer to the merchant without the need to be forwarded. If your system supports extended public keys the input address and destination will always be the same address. Anypay will provide the destination address as well as a unique nonce so you can derive the correct address.

Request:

```
{
  "destination":
"xpub68FVHpzc79cgohFThugWAQ8unhcFSjkKnuoUXxXLD1du8VvcEegUU4HpQzc5D4YmUs6udgr7E9
Lny8c2tjcmk1VN8LLcZKeD56Z8Jab8qDn",
  "nonce": 18
}
```

Respond with a pair of addresses, both of which are identical.

Response:

```
{
  "address": "XgEz3Dm9mh6VVUW6geMVagMukH3YD9Dk6D"
}
```

Forwarding Payments

When your system detects a payment it may query Anypay to find out whether there is a payment route set up for the payment's address. When there is a route set up your system is responsible for forwarding the payment. In the case that the route maintains the same address for both input and destination your system is not required to forward payments.

To query Anypay for existing routes for an address make the following request:

```
GET https://api.anypay.global/v1/routes/{address}
```

Example:

```
http

.get('https://api.anypay.global/v1/routes/XgEz3Dm9mh6VVUW6geMVagMukH3YD9Dk6D')
  .set('Authorization', 'Basic ZWx1c3VhcmlvOnlsYWNsYXZl')
```

Response:

```
{
  "input": "XgEz3Dm9mh6VVUW6geMVagMukH3YD9Dk6D",
  "destination": "XgEz3Dm9mh6VVUW6geMVagMukH3YD9Dk6D",
  "invoice_uid": "31541ada-a91d-415d-a9f4-e4bab4aeb667",
  "invoice_url": "https://api.anypay.global/invoices/31541ada-a91d-415d-a9f4-e4bab4aeb667"
}
```

The response will contain the payment destination address you need to forward the payment. Additionally the response contains a pointer to the invoice associated with this particular address.

Notifying Anypay of Payment

So far all communication has been made as HTTP requests from Anypay to your coin's service. In order for us to notify cashiers that their customers made a payment we must receive a message from your system indicating that a payment was made.

Payment Notification

Basically a payment is represented by the address of the payment, the amount paid, and the transaction id, or hash. Whenever a payment is received by an address in your system send a message back to Anypay so we can process the payment and notify the business. This message will be directly responsible for alerting customers that their payment was complete and they can walk away from the store.

Once your system receives payment and forwards the payment, post a message back to Anypay with the input payment information as well as the output payment hash.

```
POST https://api.anypay.global/v1/BSV/payments
```

Request

```
{
  "address": "XgEz3Dm9mh6VVUW6geMVagMukH3YD9Dk6D",
  "amount": 0.335,
  "hash": "8104c0e5c2d5b7318db384c5fe76eea60370f63aea37dceb1359f202506894fa",
  "currency": "AGX",
  "output_hash":
"f8fde970abfc086a94b6000e6761f31cc3ff44b763ec1ba1da029d28a29defa2"
}
```

Response

```
{
  "success": true
}
```

Authentication

You will be given an API_KEY and API_SECRET string representing authentication credentials for your coin. Provide these as HTTP headers x-api-key and x-api-secret respectively.

```
http
  .post('https://api.anypay.global/v1/payments')
  .set('Authorization', 'Basic ZWx1c3VhcmlvOnlsYWNsYXZl')
  .send({
    address: 'XgEz3Dm9mh6VVUW6geMVagMukH3YD9Dk6D'
    amount: 0.335,
    hash: '8104c0e5c2d5b7318db384c5fe76eea60370f63aea37dceb1359f202506894fa'
  });
```

Summary

Ultimately your service is responsible for making sure customer payments are delivered to the merchant's address. Anypay stores all invoice information as well as routing information from one address to the final settlement address.

Given the architecture outlined above your coin can be added into our system by simply providing a single URL to Anypay. In turn Anypay provides to your system API keys and you are ready to go.