API Introduction

The LianLian API is organized around REST. Our API has predictable resource-oriented URLs, accepts JSON-encoded request bodies, returns JSON-encoded responses, and uses standard HTTP response codes, authentication, and verbs.

Versioning

When we make backwards-incompatible changes to the API, we release new versions. The version to use is specified in the URL. The current version for the Payments API is v1 such as:

```
https://api....com/payments/v1/...
```

Authentication

The LianLian API uses an API key to authenticate requests. You will receive your `token` from LianLian Global. The `Authorization` header must be included in all requests with the following format:

```
Authorization: Basic {token}
```

Your API keys carry many privileges, so be sure to keep them secure! Do not share your secret API keys in publicly accessible areas such as GitHub, client-side code, and so forth.

All API requests must be made over HTTPS. Calls made over plain HTTP will fail. API requests without authentication will fail.

Preventing replay attacks

A replay attack is when an attacker intercepts a valid payload and its signature, then re-transmits them. To mitigate such attacks, LianLian includes a timestamp in the **LLPAY-Signature** header for requests and for responses. Because this timestamp is part of the signed payload, it is also verified by the signature, so an attacker cannot change the timestamp without invalidating the signature. If the signature is valid but the timestamp is too old, the call will be rejected.

Verifying signatures

The LLPAY-Signature header contains a timestamp and one or more signatures. The timestamp is prefixed by t=, and each signature is prefixed by a scheme. Schemes start with v, potentially followed by an integer. Currently, the only valid signature scheme is v. When the signature changes it will move to v1, v2, ...

```
LLPAY-Signature: t=1492774577,
v=5257a869e7ecebeda32affa62cdca3fa51cad7e77a0e56ff536d0ce8e108d8bd
```

Note that newlines have been added in the example above for clarity, but a real LLPAY-Signature header will be all one line.

Lianlian Pay and clients shall exchange each party's public key. The LLPAY-Signature will be signed by with the RSA private key.

Step 1: Determine the signature payload

Create a `payload` string which is the following separated by `&`

- HTTP_METHOD: The actual method defined for specific API endpoint `POST`
- URI: The request's URL Path, URI. `/api/mkt/balance`
- REQUEST_EPOCH: The seconds elapsed since 1970/1/1 00:00:00 GMT same as value of `t`
- REQUEST_PAYLOAD: The contents of request body, typically in JSON format `{"currency":"USD"}`
- QUERY_STRING: The query string of the URL, URL Encoded. `attr1%3Dvalue1%26attr2%3Dvalue2`

Put typical GET and POST examples here.

Example POST `payload` (Note there is no trailing `&` when no query string is present):

```
POST&payments/v1/merchants&19879234&{"currency":"USD"}
```

Example GET `payload`:

```
GET&payments/v1/payments/602837&19879234&&amp;currency%3DUSD
```

Step 2: Prepare the `LLPAY-Signature` string

You achieve this by concatenating:

- The timestamp (Seconds elapsed since 1970/1/1 00:00:00 GMT as a string)
- The character `,`
- Use your RSA private key and calculate the string `BASE64(RSAwithSHA256(payload))` from Step 1

Result Signature

LianLian result headers also contain the `LLPAY-Signature` header as described above but signed with Lian Lian Global's private RSA key. You may verify the signature corresponds to the Result body by calculating the signature and comparing it to the unencrypted version of `LLPAY-Signature` using the Lian Lian Global public RSA key. Here are the steps:

Step 1: Determine the signature `payload`

Create a `payload` string which is the following separated by `&`

- Response Timestamp: Extract t from LLPay-Signature and use it as timestamp
- Response Payload: The contents of response body, typically in JSON format `{"currency":"USD"}`

Example `payload`:

```
19879234&{"currency":"USD"}
```

Step 2: Use RSA Verify to compare signatures

Compare the LLPay-Signature received with the `payload` calculated in Step 1 using the Lian Lian Global Public Key using the following algorithm:

```
RSA.verify( LLPAY-Signature,
            '19879234&{" currency":"USD"}',
            Lian Lian Global Public Key )
```

Results

Success

A successful API call returns a conventional HTTP response with status 2xx indicating success. The result objects are standard REST resources. Example:

```
{
  "name": "Merchant Name",
  ...
}
```

or

```
[
  {
    "name": "Merchant Name",
    ...
  },
  {
    "name": "Merchant Name 2",
    ...
  }
]
```

Errors

LianLian uses conventional HTTP response codes to indicate the success or failure of an API request. In general: Codes in the 2xx range indicate success. Codes in the 4xx range indicate an error that failed given the information provided (e.g., a required parameter was omitted, a charge failed, etc.). Codes in the 5xx range indicate an error with LianLian's servers (these are rare).

Some 4xx errors that could be handled programmatically (e.g., a validation error) include an error code that briefly explains the error reported.

HTTP Status Table

CODE	DESCRIPTION
200	Success.
400	API Error. Detailed error codes & error message are included in each API method.
401	Unauthorized. No valid API key provided.
403	Forbidden. The API key doesn't have permissions to perform the request.
404	Resource Not Found. The requested resource doesn't exist.
500	Internal Server Error. Contact Lianlian Pay for technical support

Error Result

When a result other than 2xx is returned, an Error Object is returned with further information including Error Code and Message. There are some general error codes that can happen across APIs but most are particular to the API and defined within that API's documentation.

See definition of Error Model in API Reference.

Body

```
{
  "code": "154008",
  "message": "Client has insufficient balance to pay merchant"
 }
```

Request IDs

Each API request has an associated request identifier. You can find this value in the response headers, under `Request-Id`. If you need to contact us about a specific request, providing the request identifier will ensure the fastest possible resolution.

Argument and Field Naming

LianLian arguments and object field names follow standard underscore naming such as:

```
https://api...com/resource/?filter_by="filter"
```

```
{
  "store_name": "My Store",
  "kyc_status": "success"
}
```