baoquan Documentation

Release 1.0

baoquan.com

July 14, 2016

Contents

CHAPTER 1

Introduction

Welcome to use the Application Programming Interface of Baoquan.com- Baoquan.com API v1.0! By this API you can access to most services provided by Baoquan.com

Baoquan.com API v1.0 basically follows RESTful style. Requesting data will be transferred as JSON format, attesting data will be transferred as HTTP parameters. Processed result HTTP Status Code will demonstrate “Request success (2xx)”or “failed”. If “failed”, a detailed error message will be sent in response body as JSON format.

CHAPTER 2

Overview

For a better experience of using Baoquan.com API, you need to have a basic concept about terminology involved.

2.1 Member

Member is the signed up clients of Baoquan.com, and is also the user of Baoquan.com API. Member need to register on the Baoquan.com, and through API member can send data to Baoquan.com and realise the attestation of data.

Member could be company, organisation or personal individual.

2.2 User

User is the owner of certain attestation. User can inquiry the content of attestation by a reference number. After registered and passed through identity verification process, user is able to inquiry his/her/its attestation by his/her/its phone number and ID number.

User could be company, organisation or personal individual.

2.3 Attestation

Attestation is a piece of user data record and proof. Every attestation has a unique reference number which matches a specific owner. Through the reference number, user could inquiry the data that has been attested and apply for notarisation.

Every attestation has its linked template by its own, users could define the style of attestation by editing template.

Main data entries of attestation are: Reference number, Identity information, Template ID, Factoid set.

2.4 Identity

Identity is required when creating the attestation. Identity is the identity information of attestation owner, you can bind the identity information of attestation owner, for example the ID number, Phone number and Uniform credit code together with the attestation, so that users could easily inquiry his/her/its own attestation data. One single user could also have multiple identities.

Identity Code List:

Code	Type
ID	ID Card
MO	Mobile phone number
USCID	Uniform credit code

2.5 Template

Template is the data structure user compile on Baoquan.com by the template editor. Template provides an easy way for member to define a piece of legally bind electronic data.

2.6 Factoid

Factoid is the data segment Baoquan.com received which is related to some attestation. One attestation could only include one factoid or several factoids. Through API member could send all related factoid of an attestation once for all, or send them separately by multiple API requests.

CHAPTER 3

Request

The accessing of Baoquan.com API requires https protocol, and digital signature.

3.1 Path

The accessing of Baoquan.com API are defined as Stage environment and Formal environment. A template edited under the Formal environment could be used in the Stage environment, however, the data in Stage environment will be deleted regularly.

Baoquan.com will provide API with backward compatibility, different API versions will be denoted as strings as v1, v2.1 etc. The current version is v1.

Formal environment: *https://baoquan.com/api/v1*

Stage environment: *https://stg.baoquan.com/api/v1*

Request path = API environment + Interface name, for example: The interface of Attestation is /attestations, therefore, the request path in Formal environment is *https://baoquan.com/api/v1/attestations*

3.2 Method

All requests are HTTP POST method.

3.3 Parameters

Parameter name	Description
request_id	The id of request. This is the unique string created by accessed Member. The string length is no longer than 32 bits.
access_key	The access identifier. When you successfully uploaded a RSA public key you will receive an access_key
tonce	Request time, must be sent in Unix Time format. The difference between Tonce and server time could not beyond ±300 seconds.
payload	Signed part of data body, compiled under JSON, different request normally have different payload data.
signature	Strings singed by your RSA private key, detailed signature method will be described further later.

For example:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "access_key": "2y7cg8kmoGDrDBXJLaizoD",
        "tonce": 1464594744,
        "payload": {
                "template_id": "2hSWTZ4oqVEJKAmK2RiyT4",
        }
        "signature": "moGDrDBXJLaizoD2hSWTZ4oqVEJKAmK2RiyT4"
}
```

CHAPTER 4

Response

The response of Baoquan.com API is a JSON string. Normal response of HTTP Status Code will be 2xx, except that are error response.

4.1 Normal response

Normal response value will be shown as blow:

Field name	Description
request_id	The ID of member's request
data	Valid data, details will be described within interface.

For example:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "data": {
                "attestation_id": "nXiTrBgQ1ruCKhgZ2oV"
        }
}
```

4.2 Error response

Error response will be shown as blow

Field name	Description
request_id	The ID of member's request
message	Error causes
timestamp	Timestamp, it has millisecond level accuracy

Error HTTP Status Code

Code	Description
400	Request error (eg. Parameter not matched with format requirement)
404	Request path not existed
405	Request method error
413	Data size requested too large (eg. File uploaded too large on size)
429	Request amount beyond limitation
500	Request failed (eg. Digital signature verification failure)

For example:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "message": "",
        "timestamp": 1464672264000
}
```

CHAPTER 5

Signature

When you access Baoquan.com through API, every requested payload need to be signed by RSA private key, so that Baoquan.com cannot falsify your data.

STEP 1: Log on Baoquan.com, go into the organisation "dashboard", click the "key management"

STEP 2: Upload RSA public key

You can refer to the following shell commend to create your own certificate:

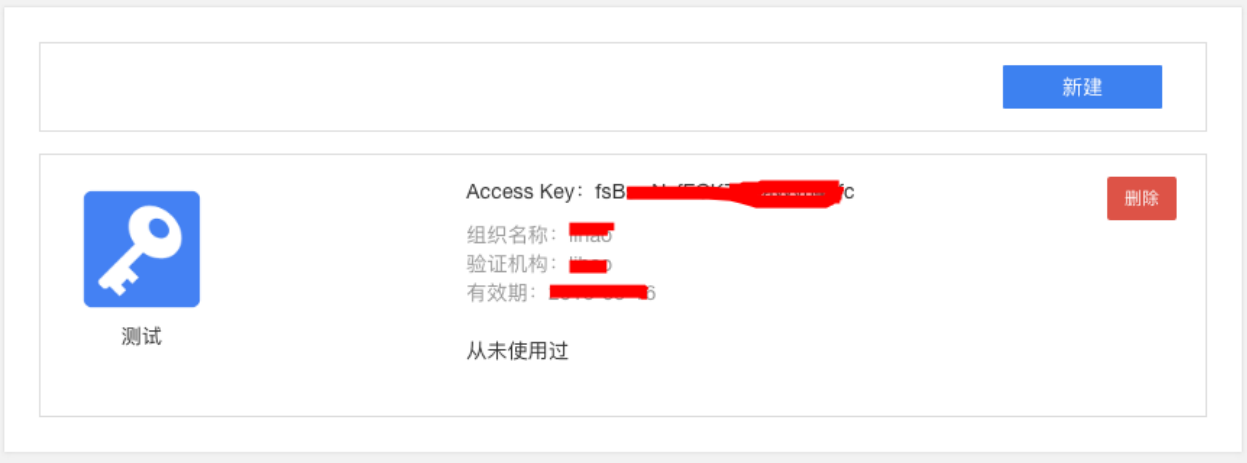
```
openssl req -x509 -newkey rsa:1024 -nodes -keyout key.pem -out cert.pem
```

Paste the content of cert.pem file to the text box, and select Sandbox or Formal.

Note: Please keep your private key safe, in addition, we strongly recommend to use the certificate issued by CA institutes as your public key under the Formal environment.

STEP 3: Construct signature

Once RSA public key has been successfully uploaded, member will obtain a Access Key as shown below:



Assume the data waiting for signature is:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "access_key": "2y7cg8kmoGDrDBXJLaizoD",
        "tonce": 1464594744,
        "payload": {
                "template_id": "2hSWTZ4oqVEJKAmK2RiyT4",
        }
}
```

The process of signing described in Java code is as the following:

```
// RSA Private Key file path
String keyFile = "/path/to/rsa_key.pem";

// Request Data
String requestId = "2XiTgZ2oVrBgGqKQ1ruCKh";
String accessKey = "2y7cg8kmoGDrDBXJLaizoD";
String tonce = 1464594744;
String payload = "{\"template_id\": \"2hSWTZ4oqVEJKAmK2RiyT4\"}";

// API path
String apiVersion = "v1";
String apiName = "attestations";
String path = String.format("/api/%s/%s", apiVersion, apiName);

// Data waiting for signature = request+API Path+requestId+accessKey+tonce+payload
String data = "".concat("POST").concat(path).concat(requestId).concat(accessKey).concat(tonce).concat

// Build signature
PEMReader pemReader = new PEMReader(new InputStreamReader(new FileInputStream(keyFile)));
PKCS8EncodedKeySpec pkcs8EncodedKeySpec = new PKCS8EncodedKeySpec(pemReader.readPemObject().getConter
pemReader.close();
KeyFactory keyFactory = KeyFactory.getInstance("RSA");
PrivateKey privateKey = keyFactory.generatePrivate(pkcs8EncodedKeySpec);
Signature signature = Signature.getInstance("SHA256WithRSA");
signature.initSign(privateKey);
signature.update(data.getBytes("UTF-8"));

// After signature encoded by Base64, the value you get is the value of signature field in requested
String signatureEncoded = Base64.getEncoder().encodeToString(signature.sign());
```

Note: The signature method applied is SHA256WithRSA, you need to used UTF-8 encode format to transform signed data from strings to bytes.

CHAPTER 6

Interface

6.1 Attestations - /attestations

When member created the template on Baoquan.com, he/she/it can send data, which is required by consummating the template, through this interface.

6.1.1 payload

Parameter name	Description	Mandatory/Optional
template_id	String, the ID of template	Mandatory
identities	Object, item for identities	Mandatory
factoids	Array, factoids set	Optional
completed	Boolean value, whether factoids set is uploaded successfully	Optional, default is true
attachments	Array, check code of attachments, optional	Optional

Factoid is an Object, which contains two fields: “type”, “data”, for example:

```
{
        "type": "hash",
        "data": {
                "userName": "David Smith",
                "idCard": "42012319800127691X"
        }
}
```

The “type” is the user defined name of factoids, “data” is the field value of factoids, as following

The template contains two factoids: "common" and "hash". attestation_no, attestation_at, product_name, organization_name in the "common" are provided by Baoquan.com when you are consummating the template. "hash" is defined by members selves, therefore, ,members need to upload that through API.

Template could contain multiple customisable factoids, for example, factoA and factoB. Members could choose to upload twice separately: upload factoA first and set "completed" to false, and then upload factoB again, set "completed" to true.

Note: Once "completed" has been set to true, no more factoids uploading will be accepted.

Assume payload is as below:

```
{
        "template_id": "2hSWTZ4oqVEJKAmK2RiyT4",
        "identities": {
                "MO": "15857112383",
                "ID": "420123198001276 91X"
        },
        "factoids": [
                {
                        "type": "hash",
                        "data": {
                                "userName": "Richard Hammond",
                                "idCard": "330124199501017791",
                                "buyAmount": 0.3,
                                "incomeStartTime": "2015-12-02",
                                "incomeEndTime": "2016-01-01",
                                "createTime": "2015-12-01 14:33:44",
                                "payTime": "2015-12-01 14:33:59",
                                "payAmount": 600
                        }
                }
        ],
        "completed": true
```

```
}
```

After Attestations, you can check the consummated content on Baoquan.com by reference number. As shown below:

6.1.2 Attachments

When you uploading the factoids data, you can upload attachments related to the factoids simultaneously. In payload, the "attachments" store the check code of attachments.

Upload a single attachment as "form" data:

```
<form method='post' enctype='multipart/form-data'>
  ...
  <input type=file name="attachments[0][]">
</form>

payload = {
        "template_id": "...",
        "identities": {...},
        "factoids": [
                {
                        "type": "...",
                        "data": {...}
                }
        ],
        "completed": true,
        "attachments": {
                "0": ["checkSum"]
        }
}
```

Upload multiple attachments as "form" data:

```
<form method='post' enctype='multipart/form-data'>
  ...
  <input type=file name="attachments[0][]">
```

```
  <input type=file name="attachments[0][]">
  <input type=file name="attachments[1][]">
</form>

payload = {
        "template_id": "...",
        "identities": {...},
        "factoids": [
                {
                        "type": "...",
                        "data": {...}
                },
                {
                        "type": "...",
                        "data": {...}
                }
        ],
        "completed": true,
        "attachments": {
                "0": ["checkSum1", "checkSum2"],
                "1": ["checkSum3"]
        }
}
```

The “key” of “attachments” is referring as the superscript of the factoids array.

The “checksum” is generated from file by SHA256, take Java as an example:

```
String file = "/path/to/file";
InputStream in = new FileInputStream(new File(file));

// Use SHA256 to hash the file
bytes[] digestBytes = DigestUtils.getDigest("SHA256").digest(StreamUtils.copyToByteArray(in));

// Transform bytes into hexadecimal
String checkSum = Hex.encodeHexString(digestBytes);
```

6.1.3 Returned data

When Attestation interface is requested successfully, the reference number will be returned.

Field name	Description
no	String, reference number of the attestation

For example:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "data": {
                "no": "rBgGqKQ1ruCKhXiTgZ2oVr",
        }
}
```

6.2 Add factoids - /factoids

Members could use Factoids interface to upload more factoids set.

6.2.1 payload

Parameter name	Description	Mandatory/Optional
ano	String, the ID of attestation	Mandatory
factoids	Array, factoids set	Mandatory
completed	Boolean value, whether factoids set is uploaded successfully	Optional, default is true
attachments	Array, checksum code of attachments, optional	Optional

For example:

```
{
        "ano": "2hSWTZ4oqVEJKAmK2RiyT4",
        "factoids": [
                {
                        "type": "hash",
                        "data": {
                                "userName": "Edward Snow",
                                "idCard": "330124199501017791",
                                "buyAmount": 0.3,
                                "incomeStartTime": "2015-12-02",
                                "incomeEndTime": "2016-01-01",
                                "createTime": "2015-12-01 14:33:44",
                                "payTime": "2015-12-01 14:33:59",
                                "payAmount": 600
                        }
                }
        ],
        "completed": false
}
```

6.2.2 Returned data

Field name	Description
success	Boolean, whether successful or not

For example:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "data": {
                "success": true,
        }
}
```

6.3 Get attestation data - /attestation

Member can get get the upload attestation data through this API, such as identities, factoids, etc.

6.3.1 payload

Parameter name	Description	Mandatory/Optional
ano	String, reference number of the attestation	Mandatory
fields	Array, returned fields	Optional, default is true

Get attestation data such as identities, factoids, attachments should take a long time to access database and decode data, so you can choose which fields you want to return.

6.3.2 Returned data

Parameter name	Description
no	Reference number of the attestation
template_id	The ID of template
identities	Identities
factoids	List of factoids
completed	Return true if factoids is uploaded, or return fasle.
attachments	List of attachments
blockchain_hash	The hash of blockchain, return null if it doesn't link to blockchain yet

attachments is an array, which The "key" of "attachments" is referring as the superscript of the factoids array, and the "value" is an attachment array.

1When fields set null, we will get all data, the result is as below:

```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "data": {
                "no": "DB0C8DB14E3C44C7B9FBBE30EB179241",
                "template_id" : "5Yhus2mVSMnQRXobRJCYgt",
                "identities": {
                        "ID": "420123198001276 91X",
                        "MO": "15857112383"
                },
                "factoids": [
                        {
                                "type": "product",
                                "data": {
                                        "name:: "zjmax",
                                        "description": "p2g financing platform""
                                }
                        },
                        {
                                "type": "user",
                                "data": {
                                        "name": "David Smith",
                                        "phone_number": "13234568732",
                                        "registered_at": "1466674609",
                                        "username": "tom"
                                }
                        }
                ],
                "completed": true,
                "attachments": {
                        "1": [
                                "2EHJQPs5j4SZpEKQXQ7r6C",
                                "2F81ZJXosNjzrPJsXKywAu"
                        ]
                },
                "blockchain_hash": "s5j4SZpEKQXQ7r6C2F81ZJXosNjzrPJsXKywAu"
        }
}
```

2When fields set an empty array, it doesn't return the values of identities, factoids and attachments, the result is as below:


```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "data": {
                "no": "DB0C8DB14E3C44C7B9FBBE30EB179241",
                "template_id" : "5Yhus2mVSMnQRXobRJCYgt",
                "identities": null,
                "factoids": null,
                "completed": true,
                "attachments": null,
                "blockchain_hash": "s5j4SZpEKQXQ7r6C2F81ZJXosNjzrPJsXKywAu"
        }
}
```


So when you want to get the hash of blockchain immediately, it's best way to set field as an empty array.

3When fields set an array, such as ["identities"], the result is as below:


```
{
        "request_id": "2XiTgZ2oVrBgGqKQ1ruCKh",
        "data": {
                "no": "DB0C8DB14E3C44C7B9FBBE30EB179241",
                "template_id" : "5Yhus2mVSMnQRXobRJCYgt",
                "identities": {
                        "ID": "42012319800127691X",
                        "MO": "15857112383"
                },
                "factoids": null,
                "completed": true,
                "attachments": null,
                "blockchain_hash": "s5j4SZpEKQXQ7r6C2F81ZJXosNjzrPJsXKywAu"
        }
}
```


6.4 Download the attestation file - /attestation/download

When member upload data to Baoquan, we should take some certain process(rendered by template) to create a Baoquan file. a Baoquan file will be eventually hashed to the blockchain, so it can eventually make a notarised certificate at notary office or make a judicial report at judicial evaluation center.

6.4.1 payload

Parameter name	Description	Mandatory/Optional
ano	String, reference number of the attestation	Mandatory

6.4.2 Returned file

This interface will get the Baoquan file and filename. The file is body of the result of http message, and the filename is header of http message. Content=Disposition is the header name. The value of header is like below:

```
form-data; name=Content-Disposition; filename=5Yhus2mVSMnQRXobRJCYgt.zip
```

As Java for example:

```
// ommit using apache http client to create http request
// closeableHttpResponse is an instance of CloseableHttpResponse
HttpEntity httpEntity = closeableHttpResponse.getEntity();
Header header = closeableHttpResponse.getFirstHeader(MIME.CONTENT_DISPOSITION);
Pattern pattern = Pattern.compile(".*filename=\"(.*)\".*");
Matcher matcher = pattern.matcher(header.getValue());
String fileName = "";
if (matcher.matches()) {
fileName = matcher.group(1);
}
FileOutputStream fileOutputStream = new FileOutputStream(fileName);
IOUtils.copy(httpEntity.getContent(), fileOutputStream)
```

CHAPTER 7

SDK

Baoquan offer Java, PHP and Python SDK, and will add more language later.

7.1 Java

If you use maven, you can add dependency as below:

```
<dependency>
    <groupId>com.baoquan</groupId>
    <artifactId>eagle-sdk</artifactId>
    <version>1.0.4</version>
</dependency>
```

If you use Gradle, you can add like following:

```
compile group: 'com.baoquan', name: 'eagle-sdk', version: '1.0.4'
```

7.1.1 Create Baoquan Client

```
BaoquanClient client = new BaoquanClient();
// set API address, like Baoquan test environment
client.setHost("https://stg.baoquan.com");
// set access key
client.setAccessKey("fsBswNzfECKZH9aWyh47fc");
// set absolute path of rsa private key file
client.setPemPath("path/to/rsa_private.pem");
```

rsa private key should begin with **—–BEGIN PRIVATE KEY—–** and end with **—–END PRIVATE KEY—–**, for example:

```
-----BEGIN PRIVATE KEY-----
MIICeAIBADANBgkqhkiG9w0BAQEFAASCAmIwggJeAgEAAoGBALS8adG98pHSLEq6
kOT6PG25GMBzpiSs1oXwnPLTOVOYarffF0xSB7nk5yxbqx5BseJNz2NxyTpeJOk8
FXEI7qTbS6oYAgyH/2HMr5Az3pKGLRdIjJQrpu3qpJkzRw82qGP2MkmVkUYeOl9B
ZEUpk1GmziwrhbD0zcJITA0mnUqnAgMBAAECgYBnetUPjLTcvrwzURxyrb95hxff
4JdIuljdOUVzVnKlJUg83JOHVBQuYBvn7thLq4uAqdJK+rQfIhX6IDeaj2WqsO7Y
d4YoVxFAlfaHIICJKur15KOXuPMpdm3ilZ0c2yCTrJ0m3Xm6mpwd4blDDSupmlj4
HEXXiInGZgwfTqONAQJBAOlX3EyvE2NvzYMh39wz11fmOi0UiyIvz0immjed4dhV
0YvPjx8Gj7XGwCkzbuNwr7tlkMTaSiYR8cw1QzV4QoECQQDGSOlgAJC8oUP2+u4H
+A83jfSLlhQ8XKAJn5Din9kBvs4eKMSjTpJiDBgA7NUAhUfCqS2/m5TiTiS3X3Ij
ZKknAkEA2iaQCQks4SvnQI9s0FuPGdhdz0ODiCSWb9+CEjkCqdQhococje7+b/0u
```

```
Ldat9uilAlfD7qX96HWiTz4EZXrXAQJBAJ+CbgMl0Ul9bcBUsoHEovEtCEn2TIcW
eEPlkldNAfSuev+2CiHZhlbLpc+wtdU6YrUNBdl7HjVDabP+W0JvqscCQQDBoUR8
Y3NUOdGRcaSgwT56tP5J1cZxg1b4vCyr+YfvcEGSBrEaxEugDUjxbON4etMVflh/
H3QNSvRf4XQ44wQO
-----END PRIVATE KEY-----
```

Other initial settings

Other settings such as API version, request ID generator. You don't configure as default:

```
// set API version
client.setVersion("v1")
// set request ID generator, generator need to implement createRequestId method in interface RequestI
client.setRequestIdGenerator(CustomRequestGenerator)

// Default SDK request ID generator
public class DefaultRequestIdGenerator implements RequestIdGenerator {

        @Override
        public String createRequestId() {
                return UUID.randomUUID().toString();
        }
}
```

When client is created, you can use client method to send request message

7.1.2 Create attestation

```
CreateAttestationPayload payload = new CreateAttestationPayload();
// set template id
payload.setTemplateId("5Yhus2mVSMnQRXobRJCYgt");
// set factoids whether upload or not, if set completed as true, then you can't append factoids
payload.setCompleted(false);
// set the owner identities, the type is defined at IdentityType
Map<IdentityType, String> identities = new HashMap<>();
identities.put(IdentityType.ID, "420123198001276 91X");
identities.put(IdentityType.MO, "15857112383");
payload.setIdentities(identities);
// add factoid list
List<Factoid> factoids = new ArrayList<>();
// add product factoid
Factoid factoid = new Factoid();
Product product = new Product();
product.setName("xxx company");
product.setDescription("p2g financing platform");
factoid.setType("product");
factoid.setData(product);
factoids.add(factoid);
// add user factoid
factoid = new Factoid();
User user = new User();
user.setName("Tom Hammond");
user.setRegistered_at("1466674609");
user.setUsername("tom");
user.setPhone_number("13452345987");
factoid.setType("user");
```

```
factoid.setData(user);
factoids.add(factoid);
payload.setFactoids(factoids);
// use client createAttestation method to create CreateAttestationResponse instance, if it can be cre
try {
        CreateAttestationResponse response = client.createAttestation(payload);
        System.out.println(response.getData().getNo());
} catch (ServerException e) {
        System.out.println(e.getMessage());
}
```

If you want to upload attachments for attestation:

```
// create three attachments, each attachment is an instance of ByteArrayBody, and ContentType must se
InputStream inputStream0 = getClass().getClassLoader().getResourceAsStream("seal.png");
ByteArrayBody byteArrayBody0 = new ByteArrayBody(IOUtils.toByteArray(inputStream0), ContentType.DEFAU
InputStream inputStream1 = getClass().getClassLoader().getResourceAsStream("seal.png");
ByteArrayBody byteArrayBody1 = new ByteArrayBody(IOUtils.toByteArray(inputStream1), ContentType.DEFAU
InputStream inputStream2 = getClass().getClassLoader().getResourceAsStream("contract.pdf");
ByteArrayBody byteArrayBody2 = new ByteArrayBody(IOUtils.toByteArray(inputStream2), ContentType.DEFAU
// create the map of attachment, which the key is factoids' superscript. For example, we set the firs
Map<String, List<ByteArrayBody>> attachments = new HashMap<>();
attachments.put("0", Collections.singletonList(byteArrayBody0));
attachments.put("1", Arrays.asList(byteArrayBody1, byteArrayBody2));
// ommit to create payload
try {
        CreateAttestationResponse response = client.createAttestation(payload, attachments);
        System.out.println(response.getData().getNo());
} catch (ServerException e) {
        System.out.println(e.getMessage());
}
```

7.1.3 Add factoids

```
AddFactoidsPayload addFactoidsPayload = new AddFactoidsPayload();
// set attestation reference number
addFactoidsPayload.setAno("7F189BBB5FA1451EA8601D0693E36FE7");
// add factoid
factoids = new ArrayList<>();
factoid = new Factoid();
User user = new User();
user.setName("Tom Hammond");
user.setRegistered_at("1466674609");
user.setUsername("tom");
user.setPhone_number("13452345987");
factoid.setType("user");
factoid.setData(user);
factoids.add(factoid);
addFactoidsPayload.setFactoids(factoids);
// use client addFactoids method to create AddFactoidsResponse instance, if it's successn then return
try {
        AddFactoidsResponse response = client.addFactoids(addFactoidsPayload);
        System.out.println(response.getData().isSuccess());
} catch (ServerException e) {
        System.out.println(e.getMessage());
}
```

adding factoids can also upload attachment for factoids, as same as just upload attachment.

7.1.4 Get attestation data

```
try {
        GetAttestationResponse response = client.getAttestation("DB0C8DB14E3C44C7B9FBBE30EB179241", r
        System.out.println(response.getData());
} catch (ServerException e) {
        System.out.println(e.getMessage());
}
```

getAttestation have two parameters. The first parameter is reference number of attestation. And the second parameter is an array, which can set to returned fields.

7.1.5 Download the attestation file

```
try {
        DownloadFile downloadFile = client.downloadAttestation("7FF4E8F6A6764CD0895146581B2B28AA");

        FileOutputStream fileOutputStream = new FileOutputStream(downloadFile.getFileName());
        IOUtils.copy(downloadFile.getFile(), fileOutputStream);
        fileOutputStream.close();
} catch (ServerException e) {
        System.out.println(e.getMessage());
}
```

7.2 PHP

If you use composer, you can add dependency as below:

```
{
  "require": {
    "baoquan/eagle-sdk": "1.0.4"
  }
}
```

7.2.1 Create Baoquan Client

```
$client = new BaoquanClient();
// set API address, like Baoquan test environment
$client->setHost('https://stg.baoquan.com');
// set access key
$client->setAccessKey('fsBswNzfECKZH9aWyh47fc');
// set absolute path of rsa private key file
$client->setPemPath('path/to/rsa_private.pem');
```

rsa private key should begin with **—–BEGIN PRIVATE KEY—–** and end with **—–END PRIVATE KEY—–**, for example:

```
-----BEGIN PRIVATE KEY-----
MIICeAIBADANBgkqhkiG9w0BAQEFAASCAmIwggJeAgEAAoGBALS8adG98pHSLEq6
kOT6PG25GMBzpiSs1oXwnPLTOVOYarffF0xSB7nk5yxbqx5BseJNz2NxyTpeJOk8
```

```
FXEI7qTbS6oYAgyH/2HMr5Az3pKGLRdIjJQrpu3qpJkzRw82qGP2MkmVkUYeOl9B
ZEUpk1GmziwrhbD0zcJITA0mnUqnAgMBAAECgYBnetUPjLTcvrwzURxyrb95hxff
4JdIuljdOUVzVnKlJUg83JOHVBQuYBvn7thLq4uAqdJK+rQfIhX6IDeaj2WqsO7Y
d4YoVxFAlfaHIICJKur15KOXuPMpdm3ilZ0c2yCTrJ0m3Xm6mpwd4blDDSupmlj4
HEXXiInGZgwfTqONAQJBAOlX3EyvE2NvzYMh39wz11fmOi0UiyIvz0immjed4dhV
0YvPjx8Gj7XGwCkzbuNwr7tlkMTaSiYR8cw1QzV4QoECQQDGSOlgAJC8oUP2+u4H
+A83jfSLlhQ8XKAJn5Din9kBvs4eKMSjTpJiDBgA7NUAhUfCqS2/m5TiTiS3X3Ij
ZKknAkEA2iaQCQks4SvnQI9s0FuPGdhdz0ODiCSWb9+CEjkCqdQhococje7+b/0u
Ldat9uilAlfD7qX96HWiTz4EZXrXAQJBAJ+CbgMl0Ul9bcBUsoHEovEtCEn2TIcW
eEPlkldNAfSuev+2CiHZhlbLpc+wtdU6YrUNBdl7HjVDabP+W0JvqscCQQDBoUR8
Y3NUOdGRcaSgwT56tP5J1cZxg1b4vCyr+YfvcEGSBrEaxEugDUjxbON4etMVflh/
H3QNSvRf4XQ44wQO
-----END PRIVATE KEY-----
```

or begin with **—–BEGIN RSA PRIVATE KEY—–** and end with **—–END RSA PRIVATE KEY—–**, for example:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC0vGnRvfKR0ixKupDk+jxtuRjAc6YkrNaF8Jzy0zlTmGq33xdM
Uge55OcsW6seQbHiTc9jcck6XiTpPBVxCO6k20uqGAIMh/9hzK+QM96Shi0XSIyU
K6bt6qSZM0cPNqhj9jJJlZFGHjpfQWRFKZNRps4sK4Ww9M3CSEwNJp1KpwIDAQAB
AoGAZ3rVD4y03L68M1Eccq2/eYcX3+CXSLpY3TlFc1ZypSVIPNyTh1QULmAb5+7Y
S6uLgKnSSvq0HyIV+iA3mo9lqrDu2HeGKFcRQJX2hyCAiSrq9eSjl7jzKXZt4pWd
HNsgk6ydJt15upqcHeG5Qw0rqZpY+BxF14iJxmYMH06jjQECQQDpV9xMrxNjb82D
Id/cM9dX5jotFIsiL89Ippo3neHYVdGLz48fBo+1xsApM27jcK+7ZZDE2komEfHM
NUM1eEKBAkEAxkjpYACQvKFD9vruB/gPN430i5YUPFygCZ+Q4p/ZAb7OHijEo06S
YgwYAOzVAIVHwqktv5uU4k4kt19yI2SpJwJBANomkAkJLOEr50CPbNBbjxnYXc9D
g4gklm/fghI5AqnUIaHKHI3u/m/9Li3WrfbopQJXw+6l/eh1ok8+BGV61wECQQCf
gm4DJdFJfW3AVLKBxKLxLQhJ9kyHFnhD5ZJXTQH0rnr/tgoh2YZWy6XPsLXVOmK1
DQXZex41Q2mz/ltCb6rHAkEAwaFEfGNzVDnRkXGkoME+erT+SdXGcYNW+Lwsq/mH
73BBkgaxGsRLoA1I8WzjeHrTFX5Yfx90DUr0X+F0OOMEDg==
-----END RSA PRIVATE KEY-----
```

Other initial settings

Other settings such as API version, request ID generator. You don't configure as default:

```
// set API version
$client->setVersion('v1');
// set request ID generator, generator need to implement createRequestId method in interface RequestI
$client->setRequestIdGenerator(CustomRequestGenerator);

// Default SDK request ID generator
class DefaultRequestIdGenerator implements RequestIdGenerator
{
    /**
     * create request id
     * @return string
     */
    function createRequestId()
    {
        return md5(uniqid("", true));
    }
}
```

When client is created, you can use client method to send request message

7.2.2 Create attestation

```
// use client createAttestation method to create CreateAttestationResponse instance, if it can be cre
try {
        $response = $client->createAttestation([
                // set template id
                'template_id'=>'5Yhus2mVSMnQRXobRJCYgt',
                // set attestation owner's identities
                'identities'=>[
                    'ID'=>'42012319800127691X',
                    'MO'=>'15857112383',
                ],
                // factoids list
                'factoids'=>[
                    // product factoid
                    [
                        'type'=>'product',
                        'data'=>[
                            'name'=>'zjmax',
                            'description'=>'p2g financing platform'
                        ]
                    ],
                    // user factoid
                    [
                        'type'=>'user',
                        'data'=>[
                            'name'=>'Tom Hammond',
                            'phone_number'=>'13234568732',
                            'registered_at'=>'1466674609',
                            'username'=>'tom'
                        ]
                    ]
                ],
                // set factoids whether upload or not, if set completed as true, then you can't appe
                'completed'=>true
                ]
        );
        echo $response['data']['no'];
} catch (ServerException $e) {
        echo $e->getMessage();
}
```

If you want to upload attachments for attestation:

```
// create three attachments, which contain one product factoid and two user factoids. The key of list
// each attachment contain resource and resource_name, and resource is an instance of php.
$attachments = [
        0=>[
            [
                'resource'=>fopen(__DIR__.'/resources/seal.png', 'r'),
                'resource_name'=>'seal.png'
            ]
        ],
        1=>[
            [
                'resource'=>fopen(__DIR__.'/resources/seal.png', 'r'),
                'resource_name'=>'seal.png'
            ],
            [
```

```
                'resource'=>fopen(__DIR__.'/resources/contract.pdf', 'r'),
                'resource_name'=>'contract.pdf'
            ]
        ]
];

// use client createAttestation method to create CreateAttestationResponse  instance, if it's success
// ommit to create payload
try {
        $response = $client->createAttestation($payload, $attachments);
        echo $response['data']['no'];
} catch (ServerException $e) {
        echo $e->getMessage();
}
```

7.2.3 Add factoids

```
try {
        $response = $this->client->addFactoids([
                // set attestation reference number
                'ano'=>'7F189BBB5FA1451EA8601D0693E36FE7',
                // list of factoids
                'factoids'=>[
                    [
                        'type'=>'user',
                        'data'=>[
                            'name'=>'Tom Hammond',
                            'phone_number'=>'13234568732',
                            'registered_at'=>'1466674609',
                            'username'=>'tom'
                        ]
                    ]
                ],
                'completed'=>true
                ]
        );
        echo $response['data']['success'];
} catch (ServerException $e) {
        echo $e->getMessage();
}
```

adding factoids can also upload attachment for factoids, as same as just upload attachment.

7.2.4 Get attestation data

```
try {
        $response = $this->client->getAttestation('DB0C8DB14E3C44C7B9FBBE30EB179241');
        var_dump($response['data']);
} catch (ServerException $e) {
        echo $e->getMessage();
}
```

getAttestation have two parameters. The first parameter is reference number of attestation. And the second parameter is an array, which can set to returned fields.

7.2.5 Download the attestation file

```
try {
        $response = $this->client->downloadAttestation('DB0C8DB14E3C44C7B9FBBE30EB179241');
        $file = fopen($response['file_name'], 'w');
        fwrite($file, $response['file']->getContents());
        fclose($file);
} catch (ServerException $e) {
        echo $e->getMessage();
}
```

response contain two fields, one is file_name, and another is file, which is an instance of \Psr\Http\Message\StreamInterface.

7.3 Python

If you use pip, you can add dependency at requirements.txt as below:

```
eagle-sdk==1.0.4
```

Note: This SDK should be running at Python 3

7.3.1 Create Baoquan Client

```
from baoquan import BaoquanClient

client = BaoquanClient()
// set API address, like Baoquan test environment
client.host = 'https://stg.baoquan.com'
// set access key
client.access_key = 'fsBswNzfECKZH9aWyh47fc'
// set absolute path of rsa private key file
client.pem_path = 'path/to/rsa_private.pem'
```

rsa private key should begin with **—–BEGIN RSA PRIVATE KEY—–** and end with **—–END RSA PRIVATE KEY—–**, for example:

```
-----BEGIN RSA PRIVATE KEY-----
MIICXgIBAAKBgQC0vGnRvfKR0ixKupDk+jxtuRjAc6YkrNaF8Jzy0zlTmGq33xdM
Uge55OcsW6seQbHiTc9jcck6XiTpPBVxCO6k20uqGAIMh/9hzK+QM96Shi0XSIyU
K6bt6qSZM0cPNqhj9jJJlZFGHjpfQWRFKZNRps4sK4Ww9M3CSEwNJp1KpwIDAQAB
AoGAZ3rVD4y03L68M1Eccq2/eYcX3+CXSLpY3TlFc1ZypSVIPNyTh1QULmAb5+7Y
S6uLgKnSSvq0HyIV+iA3mo9lqrDu2HeGKFcRQJX2hyCAiSrq9eSjl7jzKXZt4pWd
HNsgk6ydJt15upqcHeG5Qw0rqZpY+BxF14iJxmYMH06jjQECQQDpV9xMrxNjb82D
Id/cM9dX5jotFIsiL89Ippo3neHYVdGLz48fBo+1xsApM27jcK+7ZZDE2komEfHM
NUM1eEKBAkEAxkjpYACQvKFD9vruB/gPN430i5YUPFygCZ+Q4p/ZAb7OHijEo06S
YgwYAOzVAIVHwqktv5uU4k4kt19yI2SpJwJBANomkAkJLOEr50CPbNBbjxnYXc9D
g4gklm/fghI5AqnUIaHKHI3u/m/9Li3WrfbopQJXw+6l/eh1ok8+BGV61wECQQCf
gm4DJdFJfW3AVLKBxKLxLQhJ9kyHFnhD5ZJXTQH0rnr/tgoh2YZWy6XPsLXVOmK1
DQXZex41Q2mz/ltCb6rHAkEAwaFEfGNzVDnRkXGkoME+erT+SdXGcYNW+Lwsq/mH
73BBkgaxGsRLoA1I8WzjeHrTFX5Yfx90DUr0X+F0OOMEDg==
-----END RSA PRIVATE KEY-----
```

Other initial settings

Other settings such as API version, request ID generator. You don't configure as default:

```
// set API version
client.version = 'v1'
// set request ID generator, generator need to return unique String and length is less than 256
// sdk default generator is uuid.uuid4
def custom_request_id_generator():
        pass
client.request_id_generator = custom_request_id_generator
```

When client is created, you can use client method to send request message

7.3.2 Create attestation

```
// use client create_attestation method to create CreateAttestationResponse instance, if it can be cr
try:
        response = client.create_attestation({
                // set template id
                'template_id': '5Yhus2mVSMnQRXobRJCYgt',
                // set attestation owner's identities
                'identities': {
                    'ID': '42012319800127691X',
                    'MO': '15857112383'
                },
                // factoids list
                'factoids': [
                    {
                        'type': 'product',
                        'data': {
                            'name': 'zjmax',
                            'description': 'p2g financing platform'
                        }
                    },
                    {
                        'type': 'user',
                        'data': {
                            'name': 'Tom Hammond',
                            'phone_number': '13234568732',
                            'registered_at': '1466674609',
                            'username': 'tom'
                        }
                    }
                ],
                // set factoids whether upload or not, if set completed as true, then you can't apper
                'completed': true
        })
        print(response['data']['no'])
except ServerException as e:
        print(e.message)
```

If you want to upload attachments for attestation:

```
// create three attachments, which contain one product factoid and two user factoids. The key of list
// each attachment contain resource, resource_name and resource_content_type. Resource is a byte arra
attachments = {
        0: [
```

```
                {
                    'resource': open(os.path.dirname(__file__) + '/resources/seal.png', 'rb').read(),
                    'resource_name': 'seal.png',
                    'resource_content_type': 'image/png'
                }
        ],
        1: [
                {
                    'resource': open(os.path.dirname(__file__) + '/resources/seal.png', 'rb').read(),
                    'resource_name': 'seal.png',
                    'resource_content_type': 'image/png'
                },
                {
                    'resource': open(os.path.dirname(__file__) + '/resources/contract.pdf', 'rb').rea
                    'resource_name': 'contract.pdf',
                    'resource_content_type': 'application/pdf'
                }
        ]
}
// use client create_attestation method to create CreateAttestationResponse  instance, if it's succes
// ommit to create payload
try:
        response = client.create_attestation(payload, attachments)
        print(response['data']['no'])
except ServerException as e:
        print(e.message)
```

7.3.3 Add factoids

```
try:
        response = client.add_factoids({
                // set reference number of attestation
                'ano': '7F189BBB5FA1451EA8601D0693E36FE7',
                // factoids list
                'factoids': [
                    {
                        'type': 'user',
                        'data': {
                            'name': 'Tom Hammond',
                            'phone_number': '13234568732',
                            'registered_at': '1466674609',
                            'username': 'tom'
                        }
                    }
                ]
                })
        print(response['data']['success'])
except ServerException as e:
        print(e.message)
```

adding factoids can also upload attachment for factoids, as same as just upload attachment.

7.3.4 Get attestation data

```
try:
        response = client.get_attestation('DB0C8DB14E3C44C7B9FBBE30EB179241')
        print(response['data'])
except ServerException as e:
        print(e.message)
```

get_attestation have two parameters. The first parameter is reference number of attestation. And the second parameter is an array, which can set to returned fields.

7.3.5 Download the attestation file

```
try:
        response = client.download_attestation('DB0C8DB14E3C44C7B9FBBE30EB179241')
        with open(response['file_name'], 'wb') as f:
                f.write(response['file_content'])
except ServerException as e:
        print(e.message)
```

response contain two fields, one is file_name, and another is a file content, which is presented by byte format.