接口公用说明

所有接口采用HASH签名的方式来验证。在正式地址上使用的是HTTPS协议，测试地址使用的是HTTP协议。

> 正式地址：https://api.luckybao365.com
>
> 测试地址：http://api.dev.luckybao365.com

---

1. 请求格式规范

目前请求的内容的编码仅支持`UTF-8`，内容格式为`JSON`，所有公用参数都做为请求头，包含的公用请求头有:

- `Accept`：选填，接口返回数据的格式，默认为`JSON`
  - application/json
  - application/xml
- `Content-Type`：必填(有请求体时)，请求内容格式，目前固定为: `application/json; charset=utf-8`
- `X-Request-Time`：必填，发起请求时的时间戳，单位秒
- `X-Request-Nonce`：必填，随机数，每次请求必须唯一，长度在36位之内，推荐`UUID`
- `Authorization`：必填，请求的验证TOKEN，详情参照下面文档：接口签名验证说明

---

2. 接口签名验证说明

2.1. 接口密钥

接口开始对接时，会提供一个用于接口签名的ID和密钥，在本文档中约定`ApiId`为接口ID，`ApiSecret`接口签名密钥，具体使用见文档中的签名方法和验证TOKEN说明

2.2. 签名内容体

签名内容体由以下几项按顺序构成，之间以`\n`换行连接:

1. 请求方法，必须**大写**，例如：`POST`

2. 请求URL的PATH路径部分，必须以 `/` 为开头，例如： `/test/api`
3. 请求URL的QUERY查询参数部分，所有参数名称以字典升序排列，值和键之间以 `=` 连接，参数之间以 `&` 连接，需要注意的是，参数的值必须以RFC3986标准编码, 该标准中空格会被百分号编码 `%20`, 假设原参数为: `aa=100&cc=测试&bb=A B`, 转换为签名内容后: `aa=100&bb=A%20B&cc=%e6%b5%8b%e8%af%95`
4. 公用请求头里的的时间戳(`X-Request-Time`)，例如： `1503479930`
5. 公用请求头里的的随机数(`X-Request-Nonce`)，例如： `550e8400-e29b-41d4-a716-446655440000`
6. 请求的内容体，也就是请求的BODY内容，如请求没有BODY内容，比如使用 `GET` 方法时，该值为一个空字符串，假设请求体为JSON格式： `{"test1":"aaaa","test2":"bbbb"}`

由上面几个例子中的内容，我门可以得出签名内容体为：

```
POST
/test/api
aa=100&bb=A%20B&cc=%e6%b5%8b%e8%af%95
1503479930
550e8400-e29b-41d4-a716-446655440000
{"test1":"aaaa","test2":"bbbb"}
```

在本文档中约定该签名内容体为 `SignBody`。

> **注意：** 签名内容之间的连接换行符必须时 `\n`，不能是 `\r\n` 或者 `\r`。最后一项的请求体没有时，必须保留空字符串。RFC3986标准是一种常用的URL编码标准，它与RFC1738标准有些细微的区别，该标准的空格会被编码成加号 `+`，如果不清楚，请查阅链接中的文档。

2.3. 签名方法

目前接口使用的是 `HMAC_SHA1` 算法进行签名的，在本文档中约定该算法方法名为 `HMAC_SHA1`。

根据密钥计算出哈希摘要值:

```
Sign = HMAC_SHA1(SignBody, ApiSecret)
```

计算得出的 `Sign` 就为接口的签名内容，现在假设

```
ApiId = 'test123'
ApiSecret = 'SdlzXFAou5SeTfsZknH9HDOBETmkcr5G'
```

根据上面的例子的 `SignBody` 值，得到得哈希摘要值(**小写**)为:
`e129bb4cf8594381ea9cbadcb39f193987e0c54b`

2.4. 验证TOKEN

得到哈希摘要值后，最终需要将其按一定的规则拼接成一个验证TOKEN。首先将`AppId`和`Sign`以冒号`:`连接，然后使用`BASE64`进行编码，最后在编码后的字符串前加一个`Sign`前缀，之间以空格连接。

根据上面文档得到的`Sign`值，计算方式如下：

```
Token = 'Sign ' + BASE64_ENCODE('test123:e129bb4cf8594381ea9cbadcb39f193987e0c54b')
```

我们最后得出的`Token`为:

```
Sign dGVzdDEyMzplMTI5YmI0Y2Y4NTk0MzgxZWE5Y2JhZGNiMzlmMTkzOTg3ZTBjNTRi
```

得到`Token`后，添加一个名为`Authorization`请求头，将其作为该请求头的值，到此整个验证过程就结束了。

最终示例的的请求内容为：

```
POST /test/api?aa=100&cc=%e6%b5%8b%e8%af%95&bb=A%20B HTTP/1.1
Accept: application/json
Content-Type: application/json; charset=utf-8
X-Request-Time: 1503479930
X-Request-Nonce: 550e8400-e29b-41d4-a716-446655440000
Authorization: Sign dGVzdDEyMzplMTI5YmI0Y2Y4NTk0MzgxZWE5Y2JhZGNiMzlmMTkzOTg3ZTBjNTRi

{"test1":"aaaa","test2":"bbbb"}
```

---

3. 返回格式规范

请求返回的格式目前支持`JSON`和`XML`两种，默认为`JSON`，返回的编码仅支持`UTF-8`。之后所有的接口都以`JSON`格式作为示例。

如果HTTP状态码为:`2XX`，则表示请求成功，否则就为失败。成功时返回的数据对照具体的接口说明，失败时返回的格式是统一的的，主要包含以下几个值：

- `name`：错误名称代码
- `message`：错误的具体消息
- `code`：错误名称子代码，大部分情况下都为`0`，保留使用

具体示例如下:

```
{
    "name" : "ValidateException",
    "message" : "姓名不能为空",
    "code" : 0
}
```

3.1. 常用HTTP状态码

请求成功的状态码(`2XX`)：

- `200`：服务器成功返回用户请求的数据，该操作是幂等的，一般用于
- `201`：服务器新建或修改数据成功
- `202`：服务器成功表示一个请求已经进入处理队列(**异步任务**)
- `204`：服务器成功处理了请求，但没有返回任何内容

请求失败的状态码(`4XX,5XX`)：

- `400`：请求有错误
- `401`：需要身份验证或者身份验证失败
- `403`：服务器拒绝请求
- `404`：请求的资源未找到
- `405`：请求的方法有错误
- `422`：请求的数据验证出现错误
- `429`：请求的频率超过限制
- `5XX`：服务器内部发生错误

本文档中所有的接口状态码按照`HTTP 1.1`的标准来设计的，所以返回的HTTP状态码可以作为简单判断的依据，其他未列出的状态码，在某些特殊的接口或者情况下也会出现，也就不一一介绍了，可自行搜索查看。

3.2. 错误名称代码

该代码列表目前还在完善中，后续会慢慢补充，先列出些常用的。

- `BadRequest`：请求错误
- `ValidateFail`：数据验证失败
- `Unauthorized`：身份验证失败
- `Forbidden`：禁止访问
- `TooManyRequests`：请求的频率超过限制
- `NotFound`：请求的资源未找到
- `MethodNotAllowed`：请求的方法有错误
- `InternalServerError`：服务器内部错误