# 挑战应答接口

## 挑战接口

该接口用于获取挑战信息，为一随机nonce值

### 请求参数

* 请求服务：企业节点业务系统HTTP端口（默认 3000）
  + 请求方式：GET
  + 请求路径：/api/v1/doip/public/challenge
  + Header:

| **参数** | **类型** | **是否必填** | **最大长度** | **备注** | **示例值** |
| --- | --- | --- | --- | --- | --- |
| Content-Type | String | 是 | - | application/json | application/json |

* param:

| **参数** | **类型** | **是否必填** | **最大长度** | **备注** | **示例值** |
| --- | --- | --- | --- | --- | --- |
| handle | String | 是 | 255 | 要查询的标识 | 88.111.911/app |

### 请求示例

/api/v1/doip/public/challenge?handle=88.111.911/app

### 响应参数

| 名称 | 类型 | 必选 | 约束 | 中文名 | 说明 |
| --- | --- | --- | --- | --- | --- |
| code | integer | true | none | none | 返回码 |
| message | string | true | none | none | 返回消息 |
| data | object | true | none | none | 返回数据 |

### 响应示例

{  
 "code": 1,  
 "message": "operation was successfully processed",  
 "data": "1816"  
}

### 异常示例

{  
 "message": "标识格式异常",  
 "code": 40508  
}

* 常见错误见附录

### 注意事项

1. 该接口只允许应用身份调用，其他身份调用无法获取nonce值。

## 应答，颁发Token

该接口用于验证使用nonce值生成的signature，验证成功则颁发token

### 请求参数

* 请求服务：企业节点业务系统HTTP端口（默认 3000）
* 请求方式: POST
* 请求路径: /api/v1/doip/public/verify-response
* Header:

| **参数** | **类型** | **是否必填** | **最大长度** | **备注** | **示例值** |
| --- | --- | --- | --- | --- | --- |
| Content-Type | String | 是 | - | application/json | application/json |

* Body:

| 名称 | 位置 | 类型 | 必选 | 说明 |
| --- | --- | --- | --- | --- |
| handle | body | string | 是 | 标识 |
| signature | body | string | 是 | 签名 |

### 返回数据结构

| 名称 | 类型 | 必选 | 约束 | 中文名 | 说明 |
| --- | --- | --- | --- | --- | --- |
| code | integer | true | none | none | 返回码 |
| message | string | true | none | none | 返回消息 |
| data | object | true | none | none |  |
| data.token | string | true | none | none | 返回token |

### 响应示例

{  
 "code": 1,  
 "message": "operation was successfully processed",  
 "data": {  
 "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiI4OC42MDguODg4L0FwcF96ajAxIiwiZXhwIjoxNzAyNTQyMTM0LCJpYXQiOjE3MDI0NTU3MzQsImp0aSI6IjMxMzlmNDVkLTA5YzAtNDljZS04MTdkLWE4NDRhZmFlZTFjMyJ9.9Akw3pAwMM-tHZ6WOB6OI4E\_ZMvDKyRn6CPYD1TWahc"  
 }  
}

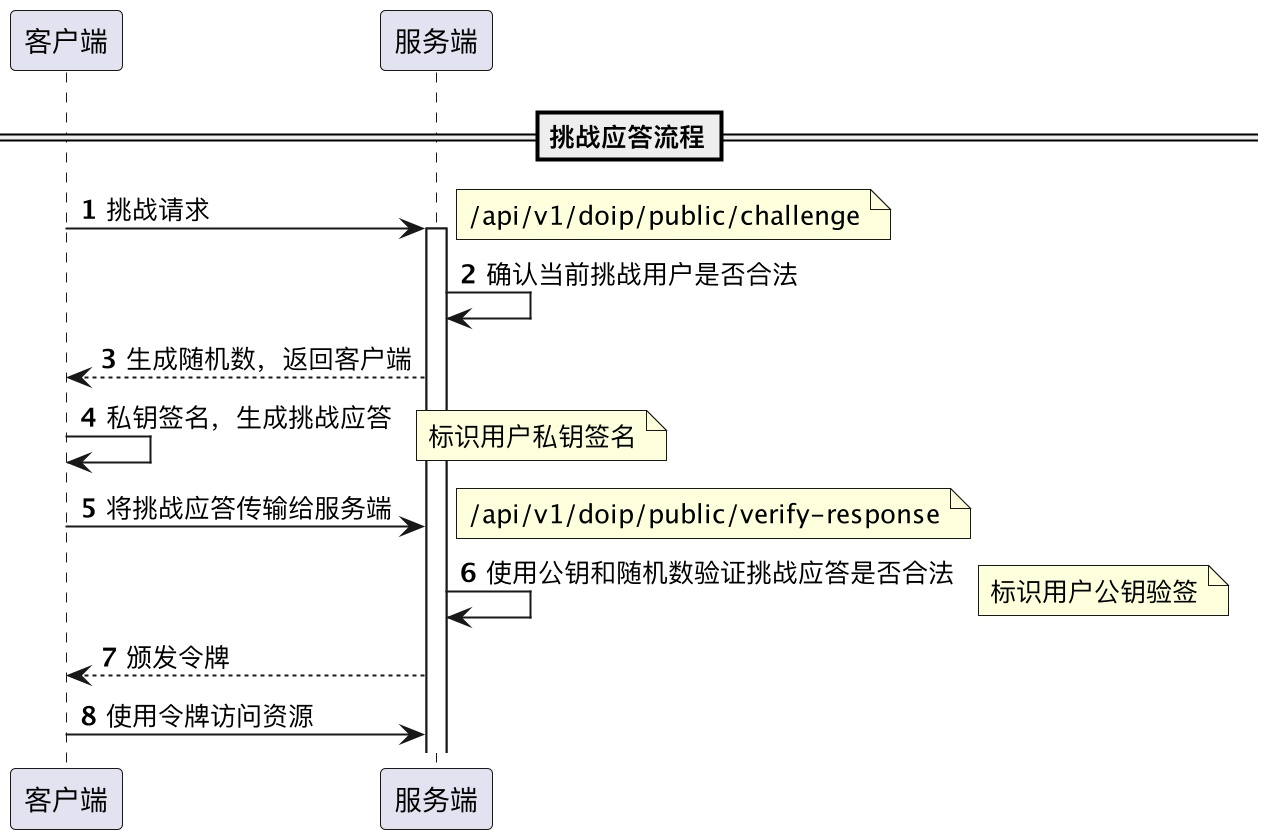
### 异常示例

{  
 "message": "标识格式异常",  
 "code": 40508  
}

## 签名生成方式参考

### 企业节点接口认证流程

企业节点OpenApi认证使用挑战应答的方式，主体流程如下



img.png

### 签名生成

#### Java

/\*\*  
 \* 私钥签名  
 \* @param data  
 \* @param privateKey  
 \* @return  
 \* @throws Exception  
 \*/  
public static String sign(byte[] data, PrivateKey privateKey) throws Exception {  
 // 用私钥对信息生成数字签名  
 Signature signature = Signature.getInstance(SIGNATURE\_ALGORITHM);  
 signature.initSign(privateKey);  
 signature.update(data);  
  
 return Base64.getEncoder().encodeToString(signature.sign());  
 }  
  
/\*\*  
 \* 转换私钥格式  
 \*/  
public static PrivateKey fromPkcs8Pem(String pem) throws Exception {  
 return fromPkcs8Pem(pem, null);  
 }  
  
/\*\*  
 \* 转换私钥格式  
 \*/  
public static PrivateKey fromPkcs8Pem(String pem, String passphrase) throws Exception {  
 BytesAndKeyType bytesAndKeyType = readPemFile(new StringReader(pem));  
 boolean encrypted = "ENCRYPTED PRIVATE".equals(bytesAndKeyType.keyType);  
 if (!encrypted && !"PRIVATE".equals(bytesAndKeyType.keyType)) {  
 throw new Exception("Expected -----BEGIN [ENCRYPTED] PRIVATE KEY-----");  
 }  
 return privateKeyFromBytes(bytesAndKeyType.bytes, encrypted, passphrase);  
 }

### Python

#!/usr/bin/env python  
#coding=utf-8  
import os  
import json  
import base64  
# RSA  
from Crypto.PublicKey import RSA  
from Crypto.Signature import PKCS1\_v1\_5  
from Crypto.Hash import SHA256  
   
# admin1-app1  
path\_to\_private\_key\_pem\_file = '//Users//graypig//Downloads//App\_app100.pem'  
  
nonce="1850" ## 替换为challenge接口获取的随机数  
byte\_array\_nonce = nonce.encode()  
   
def sign\_bytes\_rsa( byte\_array\_nonce, path\_to\_private\_key\_pem\_file):  
 key = open(path\_to\_private\_key\_pem\_file, 'r').read()  
 rsa\_key = RSA.importKey(key)  
 signer = PKCS1\_v1\_5.new(rsa\_key)  
 \_digest = SHA256.new()  
 #print len(nonce)  
 # print len(digest)  
 \_digest.update(byte\_array\_nonce)  
 # \_digest.update(digest)  
 sign = signer.sign(\_digest)  
 #print len(sign)  
 return base64.b64encode(sign)  
   
sign = sign\_bytes\_rsa( byte\_array\_nonce, path\_to\_private\_key\_pem\_file)  
print(sign)

``````