API 使用文档（angular 版）

准备工作

请确保你已经获得了 clientID 和 clientSecret，要使用 api，它们是必须的。

如果你还没有获得它们，请参考：
https://sandbox.developerhub.citi.com/get-started#guide-title-705

1、用户登录

1.1 获取客户端凭证

1.1.1 接口功能说明

本接口主要用于获取客户端凭证 access_token，此 access_token 可以用于用户登录的进一步认证。

1.1.2 详细参数说明

Url: https://sandbox.apihub.citi.com/gcb/api/clientCredentials/oauth2/token/hk/gcb

（注：本文档均基于 Hong Kong Api 进行说明。）

请求方式： POST

请求头（requestHeader）所需包含参数：

```
{
    'content-type': 'application/x-www-form-urlencoded',
    authorization: "Basic " + encryptedKey,
    accept: 'application/json'
}
```

Aauthorization：字符串，其中 ***encryptedKey*** 为 clientID clientSecret 通过冒号’:’拼接出的字符串进行 base64 加密后的密文，注意最后需要加上前缀 ″Basic ″。加密过程具体如下图；

```
const clientID = "f2bdc852-3396-4b87-9e2e-4ef3ff83a983";
const clientSecret = "dH2eR3qC8gC5uX7mE7cQ4dR5vU3wC8iD7nN1uI2uS6qJ1cX8xF";
const combinePlainTxt = clientID + ":" + clientSecret;
const encryptedKey = window.btoa(decodeURIComponent(encodeURIComponent(combinePlainTxt)));
```

请求参数：

▼ Form Data　view source　view URL encoded

grant_type: client_credentials

scope: /api

返回参数：


```
{
    "access_token": "AAEkYzFjMDQ0Y2UtNTBmMy00NmY4LWI4YjEtYmQ5ODJkMWZiNGZh3xGP85xjqyxoHR7pXxzQJf223kWPL-HyWHD4zrRCvHZUkeBkTgxppbm
    "refresh_token": "AAGsyASCzlBplxGvA-5CFCkLhNinu6-0HQt-y7PuzsRLVAHok6yYs6KS2Np4t7bL0R8FMeT62wYXFxxY6F7LU_cc00QTXPfoQFFtay2tu3eGpBAGDg
    "scope": "/dda/customer /dda/accountlist /dda/account /dda/accountsdetails /dda/account/transactions",
    "token_type": "bearer",
    "expires_in": 1800
}
```


1.1.3 调用实例

定义服务：

```
loginStep1(): Promise<any> {
    const clientID = "f2bdc852-3396-4b87-9e2e-4ef3ff83a983";
    const clientSecret = "dH2eR3qC8gC5uX7mE7cQ4dR5vU3wC8iD7nN1uI2uS6qJ1cX8xF";
    const combinePlainTxt = clientID + ":" + clientSecret;
    const encryptedKey = window.btoa(decodeURIComponent(encodeURIComponent(combinePlainTxt)));
    let url = 'https://sandbox.apihub.citi.com/gcb/api/clientCredentials/oauth2/token/hk/gcb';
    let params = "grant_type=client_credentials&scope=/api";

    let httpOptions = {
        headers: new HttpHeaders(
            {
                'content-type': 'application/x-www-form-urlencoded',
                authorization: "Basic " + encryptedKey,
                accept: 'application/json'
            }
        )
    };

    return this.http.post(url, params
        , httpOptions)
        .toPromise()
        .then(response => response)
        .catch(this.handleError);
}
```

调用服务：

```
this.loginService.loginStep1()
    .then((res) => {
        //do something
        return Promise.resolve(res.access_token);
    })
```

1.2 获取加密算法参数

1.2.1 接口功能说明

本接口主要基于 1.1 接口中获取到的 access_token 进行请求，并获取到 modulus 和 exponent 参数用于之后得 RSA 密码加密。

1.2.2 详细参数说明

Url: https://sandbox.apihub.citi.com/gcb/api/security/e2eKey

请求方式： GET

请求头所需包含参数：

```
{
    'content-type': 'application/json',
    authorization: "Bearer " + accessToken,
    client_id: "f2bdc852-3396-4b87-9e2e-4ef3ff83a983",
    uuid: uuid
}
```

authorization： 参数中包含的 ***accessToken*** 为 1.1 接口中 response 所返回的 access_token。

client_id: 你所创建的 app 所对应的 client_id。

uuid: 你需要每次生成不同的 uuid 来进行请求。

请求参数：

无

返回参数：


```
{
   "modulus": "9e698de123c3484b174dcaadbc4cb46bc31341a2b743a1a25a1a34e44c9b655f31bfded01b9e14a9b548511d1bfdb9d93b2bbd8fe029fea49376e2f652f
   "exponent": "10001"
}
```


modulus 和 exponent: 这些都是之后使用 RSA 算法加密登陆密码时所必须的参数。

返回头（responseHeader）：

```
bizToken: jv7pa3BI3a8qzg1VhKJ0c3i3AhxCPYSOwiVaSJIXypQgGUQ172XHusmZ4DQX8n9LjzscBuM5Z9J9JgIM06m7ng==
citiuuid: 8a548e5c-9cb0-4dda-b5f1-c49827bed9c5
Connection: keep-alive
Content-Length: 548
Content-Type: application/json
Date: Mon, 09 Jul 2018 07:15:13 GMT
eventid: 24FCE4D8FA4E6E1212E7196060852307
```

bizToken：登陆请求时所必须的参数。

eventid：和刚刚提到的 modulus 和 exponent 一样是 RSA 加密所需的参数。

2.1.3 调用实例

定义服务：

```
loginStep2(accessToken): Observable<HttpResponse<any>> {
    let uuid = UUID.UUID();
    return this.http.get<any>(this.url
        , {
            headers: {
                'content-type': 'application/json',
                authorization: "Bearer " + accessToken,
                client_id: "f2bdc852-3396-4b87-9e2e-4ef3ff83a983",
                uuid: uuid
            }, observe: 'response'
        });
}
```

注意：因为需要获取 responseHeader 中所包含的参数，所以需要返回 HttpResponse 类型的可观察对象，并在请求 option 中包含 observe：‘response’以订阅响应相关参数。

调用服务：

```
this.loginService.loginStep2(res).subscribe(resp => {
    // display its headers
    console.log("resp param"+resp.body.modulus);
    console.log("resp header param"+resp.headers.get('bizToken'));
});
```

调用此服务后，可通过 resp.body.xxx 获取对应的响应参数，通过 resp.headers.get(‘propertyName’)获取对应响应头参数。

1.3 用户账户名密码登录

1.3.1 接口功能说明

本接口利用 1.2 接口所获取的响应参数及响应头参数，对用户输入的密码进行 RSA 算法加密，并最终请求登陆许可。

1.3.2 详细参数说明

Url: https://sandbox.apihub.citi.com/gcb/api/password/oauth2/token/hk/gcb

请求方式： POST

请求头所需包含参数：

```
{
    'content-type': 'application/x-www-form-urlencoded',
    authorization: "Basic " + encryptedKey,
    accept: 'application/json',
    uuid: uuid,
    bizToken: bizToken
}
```

authorization: 此处同 1.1 的 authorization。

uuid: 你需要每次生成不同的 uuid 来进行请求。

bizToken: 1.2 接口中 responseHeader 所返回的 bizToken。

请求参数：

```
grant_type: password
scope: /api
username: SandboxUser1
password: 91b667e126174b53eaa59691b9b92816f7322dc26eb362aa9ae9435247950a6a3b106514df12cede4ec842c3487b5686c01c1a746c88ba0b410f939f0bbb862d0
cf053a09a945c8cee0d296a5d070ce08ec6ed2428f4bc8875703773f5e98ef331f35b83348583e4632a0965427ec963c5e0a8bca98de79bd3c16b03daa4485dbb549ba42996
cc66876a3ca60d
```

返回参数：


```
{
  "token_type": "bearer",
  "access_token": "AAIkZjJiZGM4NTItMzM5Ni00Yjg3LTllMmUtNGVmM2ZmODNhOTgz-i-53Cx94rqg0nlZlYMccghGBi5ph0xG-ASWCh
  "expires_in": 1800,
  "consented_on": 1531121451,
  "scope": "/api",
  "refresh_token": "AAJqE9grkNbWnvXKHkNB-eDmJ-sOelZau9Q9IFRp6Ihx5PCPMJ9NvnmVo_QzZe1BT1afhTx_pTPP4v8jrrpEqOTM-YK
  "refresh_token_expires_in": 2592000
}
```


access_token: 此 access_token 为用户登录后进行接口的请求的必须参数。获取此 access_token 视为登陆成功，可进行后续操作。

3.1.3 调用实例

定义服务：

```
loginStep3(bizToken, username, password, eventid, modulus, exponent): Promise<any> {
    this.encryptPwd(password,eventid,modulus,exponent);
    let params = "grant_type=password&scope=/api&username=" + username + "&password=" +
    this.encryptPwd(password,eventid,modulus,exponent)
    let uuid = UUID.UUID();
    let options = {
        headers: new HttpHeaders({
            'content-type': 'application/x-www-form-urlencoded',
            authorization: "Basic " + encryptedKey,
            accept: 'application/json',
            uuid: uuid,
            bizToken: bizToken
        })
    }
    return this.http.post(this.url, params
        , options)
        .toPromise()
        .then(response => response)
        .catch(this.handleError);
}
```

其中 encryptPwd 函数基于 eventid, modulus, exponent 对用户输入的 password 进行 RSA 加密，具体 RSA 加密工具请自行选择或使用：
https://sandbox.developerhub.citi.com/sites/sandbox.developerhub.citi.com/modules/custom/citi_apic_oauth/CitiE2E.js

如果选择引入 citiE2E.js 则范例代码如下：

```
encryptPwd(password, eventid, modulus, exponent): string {
    var pub = new RSAKey();
    pub.setPublic(modulus, exponent);
    var encrypted_password;
    var unencrypted_data = eventid + ",b" + password;
    encrypted_password = pub.encryptB(getByteArray(unencrypted_data)).toString(16);
    return encrypted_password;
}
```

调用服务：

```
this.loginService.loginStep3(bizToken, this.username, this.password, eventid, modulus, exponent).then((res) => {
    //do something
})
```

2. 获取账户总览

2.1 接口功能说明

本接口获取登陆用户的 account 信息总览数据。

2.2 详细参数说明

Url: https://sandbox.apihub.citi.com/gcb/api/v1/accounts

请求方式：GET

请求头所需包含参数：

```
{
    'content-type': 'application/json',
    authorization: "Bearer " + accessToken,
    accept: 'application/json',
    client_id: "f2bdc852-3396-4b87-9e2e-4ef3ff83a983",
    uuid: uuid
}
```

authorization: 此处所包含的 accessToken 为 1.3 接口中所获取的 access_token。

client_id: 你所创建的 app 所对应的 client_id。

uuid: 你需要每次生成不同的 uuid 来进行请求。

请求参数：

无

返回参数：


```
{
  "accountGroupSummary": [{
    "accountGroup": "SAVINGS_AND_INVESTMENTS",
    "accounts": [{
      "savingsAccountSummary": {
        "productName": "Personal Savings Account",
        "productCode": "0004_SPSAV",
        "accountNickname": "PERSAVING",
        "displayAccountNumber": "XXXXXX1847",
        "accountId": "355a515030616a53576b6a65797359506a634175764a734a3238314e4668627349486a676f7449463949453d",
        "currencyCode": "HKD",
        "accountClassification": "ASSET",
        "accountStatus": "ACTIVE",
        "currentBalance": 13642.83,
        "availableBalance": 13642.83
      }
```


2.3 调用实例

定义服务：

```
getAccountSummary(accessToken): Promise<any> {
    let uuid = UUID.UUID();
    let httpOptions = {
        headers: new HttpHeaders({
            'content-type': 'application/json',
            authorization: "Bearer " + accessToken,
            accept: 'application/json',
            client_id: "f2bdc852-3396-4b87-9e2e-4ef3ff83a983",
            uuid: uuid
        })
    };
    return this.http.get(this.url, httpOptions)
        .toPromise()
        .then(response => response)
        .catch(this.handleError);
}
```

调用服务：

```
this.accountSummaryService.getAccountSummary(this.accessToken).then((res) => {
    //do something
})
```

3. FAQ

3.1 如何引入第三方 js 和 css 并使用？

在 angular-cli.json 中，在 styles 和 scripts 属性中可以引入第三方 js 和 css，注意：此处引用为全局引用。

```
"styles": [
  "styles.css",
  "../node_modules/bootstrap/dist/css/bootstrap.min.css",
  "../node_modules/font-awesome/css/font-awesome.min.css"
],
"scripts": [
  "../node_modules/jquery/dist/jquery.min.js",
  "../node_modules/jquery/dist/jquery.min.js",
  "../node_modules/bootstrap/dist/js/bootstrap.min.js",
  "../src/assets/js/CitiE2E.js"

],
```

3.2 如何获取请求的响应头中的信息？

定义服务时，定义返回类型为 HttpResponse 的可观察对象（Observable），并在请求的 options 中增加 observe: ‘response’，这样就可以在调用服务时，通过 response.headers 获取响应头的相关信息

```
loginStep2(accessToken): Observable<HttpResponse<any>> {
    let uuid = UUID.UUID();
    return this.http.get<any>(this.url
        , {
            headers: {
                'content-type': 'application/json',
                authorization: "Bearer " + accessToken,
                client_id: "f2bdc852-3396-4b87-9e2e-4ef3ff83a983",
                uuid: uuid
            }, observe: 'response'
        });
}
```

```
this.loginService.loginStep2(res).subscribe(resp => {
    // display its headers
    console.log("resp param"+resp.body.modulus);
    console.log("resp header param"+resp.headers.get('bizToken'));
});
```

3.3 为何经常在页面渲染时遇到 Cannot read property 'some-property-name' of undefined?

在模板中绑定组件变量时，有时会需要绑定较深的对象层级，比如 a.b.c.d.e。如果初始化时 a 是空值或者 undefined 就会出现上述错误，避免错误的方法是在外层元素加上*ngIf=”判定 a 不为空”，有时会需要判定 a.b.c 不为空，视具体情况而定。