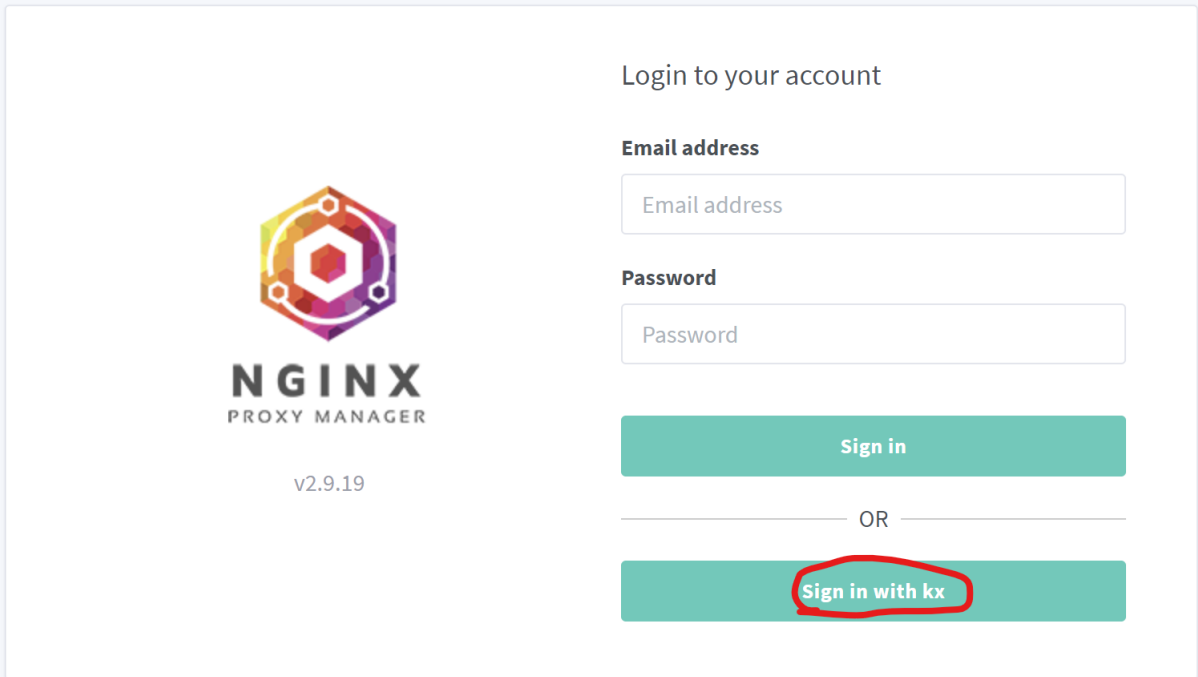


OpenId 介接說明

介接成功圖片範例：

以Nginx為例：

Step1：進入服務系統，會跳出需要登入介面，點選SSO按鈕



Login to your account

Email address

Email address

Password

Password

Sign in

OR

Sign in with kx

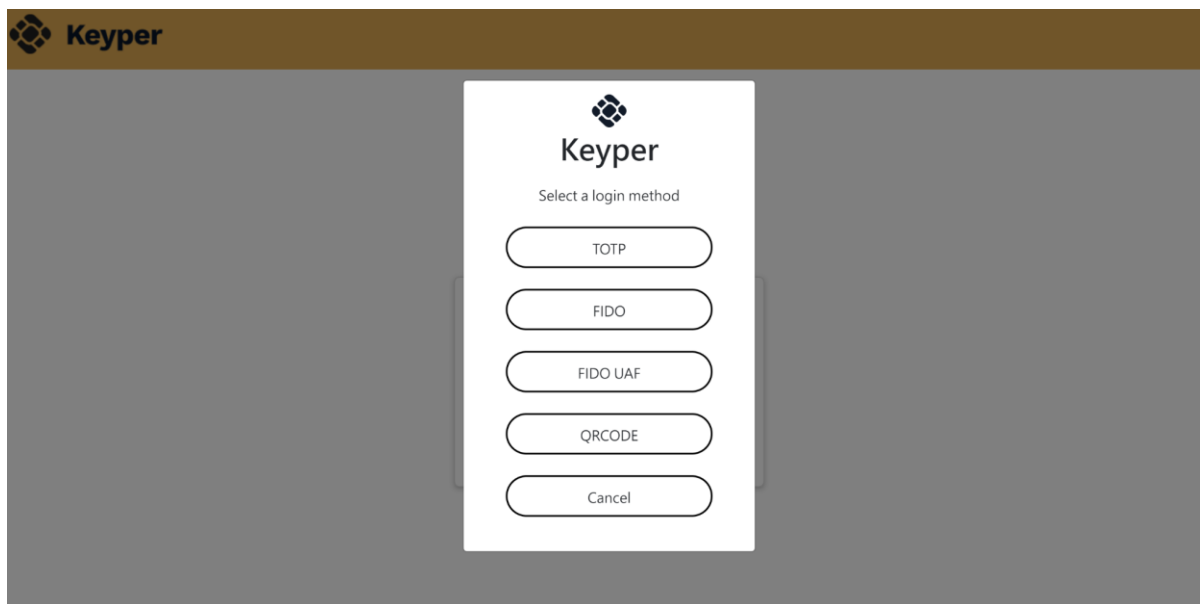
Step2：進入keyper，進行身分驗證登入

A small, white, rounded rectangular dialog box with a thin gray border. At the top, it says 'Sign in With Keyper' in bold black text, followed by a short orange horizontal line. Below this is a text input field with the placeholder text 'Enter Account'. At the bottom of the dialog is an orange button with the word 'Continue' in white text.

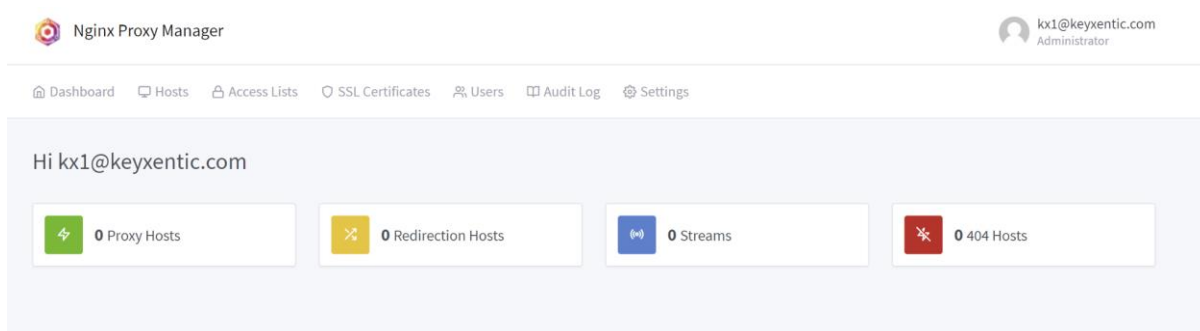
Step3：輸入帳號

The main sign-in screen for Keyper. It has a white background with a large, bold, dark blue title 'Sign in With Keyper' at the top, followed by a short orange horizontal line. Below the title is a large, light blue rounded rectangular input field with a blue border. Inside this field, the email address 'kx1@keyxentic.com' is entered, followed by a vertical cursor. At the bottom of the screen is a large orange button with the word 'Continue' in white text.

Step4：跳出可驗證方式



Step5：進入服務系統



介接流程說明

以下說明

SP(service provider)為客戶服務端

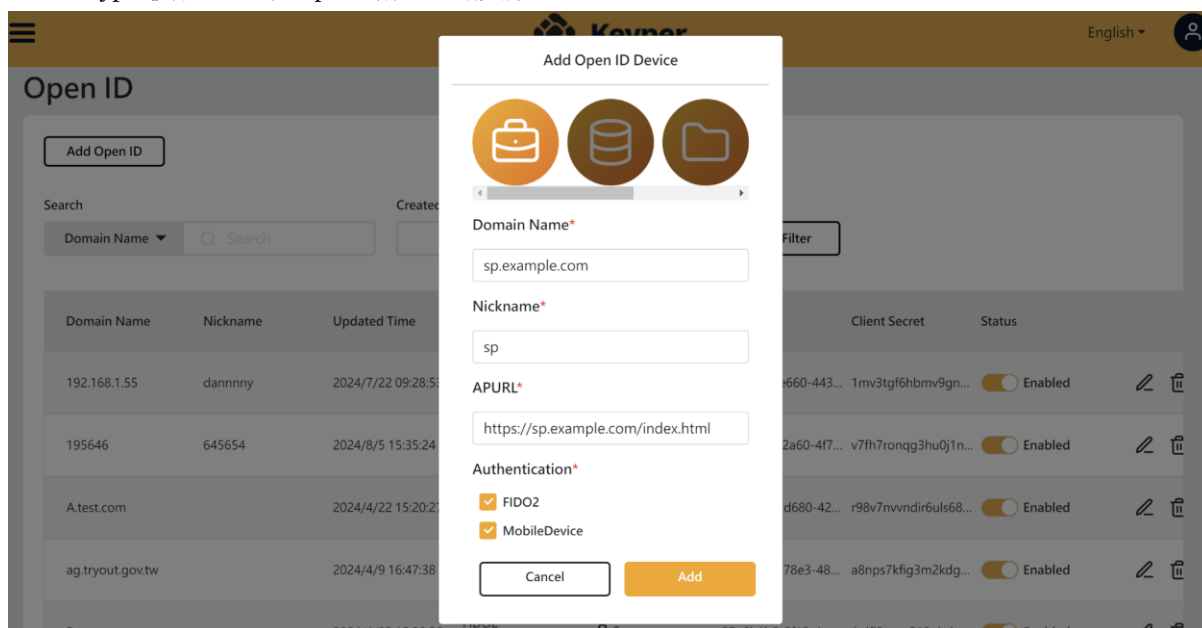
IDP(identity provider)為Keyper身分認證

情境：

User進入sp無token(無法識別身分)的情況下，由SP進行以下OpenId步驟取得id_token(JWT)格式並解析後即可取得User資訊。

Step1:於Keyper Server 加入OpenId，取得ClientId & client secret

1. 登入Keyper後台，並於OpenId加入SP服務



Domain Name: 輸入 SP Domain

Nickname: 輸入SP別稱

APURL：輸入SP 首頁URL

2. 新增後取得SP client ID & client Secret

Domain Name	Nickname	Updated Time	Authentication	Using Member	Client ID	Client Secret	Status		
mistest.keyxentic....		2024/5/9 15:07:36	FIDO2 MobileDevice	0	ea6a4dae-d13a-40...	1aapaf2cm02qkrip...	<input checked="" type="checkbox"/> Enabled		
portal.azure.com	azure	2024/8/5 14:02:14	FIDO2 MobileDevice	0	21ea775a-ed63-46f...	nfqmh82a2fu2ulcfu...	<input checked="" type="checkbox"/> Enabled		
sp.example.com	sp	null	FIDO2 MobileDevice	0	<u>66be5fb2-4780-47...</u>	<u>biv8j83pilj7cak4r5n...</u>	<input checked="" type="checkbox"/> Enabled		
sso.com	test	null	FIDO2	0	1b5c5d20-f6eb-46...	6cnihs08582o5g8rr...	<input type="checkbox"/> Disabled		

3. 自行開啟該domain及授權group設定。

< Open ID

Domain

Nickname

APURL

sp.example.com

sp

https://sp.example.com/inc

m

Status ☒ Enabled

Enabled Whitelist by Group: ☒ Enabled

Authentication: ☒ MobileDevice ☒ FIDO2

Back

Save

Add Group

Search

Group Name

Clear Filter

Step2:取得Keyper OpenId Configuration資訊

API :

GetOpenIdConfiguration

Authentication : None

Method : GET

Service path : [Keyper Server URL]/KeyperManagement/.well-known/openid-configuration

提示: Keyper Server URL: 根據Domain填寫 Keyper Server URL 。 ex. https://keyper-test.keyxentic.com:8443

Service Input :

format : JSON

Name	Type	Required	Description

Service Output :

format : JSON

Parameter:

Name	Type	Description
issuer	String	驗JWT iss欄位
authorization_endpoint	String	發起openId auth URL
token_endpoint	String	發起 auth code 換 token 的URL
revocation_endpoint	String	登出 URL
response_types_supported	String	
subject_types_supported	String	
id_token_signing_alg_values_supported	String	
jwks_uri	String	取得驗簽JWT key 的 URL
scopes_supported	String	可帶入的 scopes
claims_supported	String	JWT claims 全部會帶的資訊(部分需依照scope給)
grant_types_supported	String	grant type 所支持的項目

Example :

```
{
  "response_types_supported": [
    "code",
    "id_token",
    "id_token token"
  ],
  "claims_supported": [
    "aud",
    "email",
    "email_verified",
    "iss",
    "sub",
    "exp",
    "iat",
    "name"
  ],
  "jwks_uri": "https://keyperdev.keyxentic.com:8443/KeyperManagement/keyper/service/OpenId/certs",
  "grant_types_supported": [
    "authorization_code",
    "refresh_token"
  ],
  "subject_types_supported": [
    "public"
  ],
  "revocation_endpoint": "https://keyperdev.keyxentic.com:8443/KeyperManagement/keyper/service/OpenId/logout",
  "id_token_signing_alg_values_supported": [
    "RS256"
  ],
  "scopes_supported": [
    "openid",
    "email",
    "profile"
  ],
  "issuer": "https://keyperdev.keyxentic.com:8443/KeyperManagement",
  "authorization_endpoint": "https://keyperdev.keyxentic.com:8443/KeyperManagement/keyper/service/OpenId/auth",
  "token_endpoint": "https://keyperdev.keyxentic.com:8443/KeyperManagement/keyper/service/OpenId/token"
}
```

Demo Code :

```
String openIdInfo = Methods.doGet(urlConst.getKeyperServer() + ".well-known/openid-configuration");
public static String doGet(String targetUrl) throws IOException {
    URL url = new URL(targetUrl);

    // Open connection
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();

    // Set request method to POST
    connection.setRequestMethod("GET");

    // Enable input/output streams
    connection.setDoOutput(false);
    connection.setDoInput(true);

    // Set request headers
    connection.setRequestProperty("Content-Type", "application/json");

    // Get response code
    int responseCode = connection.getResponseCode();

    StringBuilder response = new StringBuilder();
    // Read response data
    try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
        String line;
        while ((line = reader.readLine()) != null) {
            response.append(line);
        }
    }
    logger.info(response.toString());

    // Close connection
    connection.disconnect();
    return response.toString();
}
```


Step 3:發起Open ID Auth請求

API :

Auth

Authentication : None

Method : Get

Service path : [Keyper Server URL]/KeyperManagement/keyper/service/OpenId/auth

Service Input :

Content-Type : application/x-www-form-urlencoded

format : parameter

Name	Type	Required	Description
client_id	string	yes	註冊時分配的clientId
redirect_uri	string	yes	完成登入流程後，將code 與 state 回傳給此URL
response_type		yes	Value MUST be set to "code".
scope		yes	範圍參數可加入 profile , email 。 ex:openId profile
state	string	no	sp所亂數生成，此參數會與 response時 一併回復供驗證
nonce	string	no	sp所亂數生成，此參數會於Id token一併回復供驗證。

Service Output :

format : redirect to loginPage

驗證完成後

系統會將response傳送到您在request指定的redirect_uri。所有回應都會在查詢字串中傳回

response example:

<https://openid.example.com/callback?code=rBcHMNOVcKiPfrNuHbdNyrKRLsBGSPOMZmtIVwwBjRpiGNJZGPzJIwyHogPPYdsOvjHsGYozG&state=3a17a915-aed9-46b6-b777-7607e107f9e8>

Demo Code :

```
Map<String, String> requestDate = new HashMap<>();
requestDate.put(Const.STRING_CLIENTID, "your clientId" );
requestDate.put(Const.STRING_RESPONSE_TYPE, "code");
requestDate.put(Const.STRING_SCOPE, "openid profile");
requestDate.put(Const.STRING_REDIRECT_URI, "sp server redirect uri" );
if (openIdConst.isState()){
    requestDate.put(Const.STRING_STATE, UUID.randomUUID().toString());
}
if (openIdConst.isNonce()){
    requestDate.put(Const.STRING_NONCE, UUID.randomUUID().toString());
}

res.sendRedirect( "oeprId auth endpoint" + "?" + Methods.mapToUrlParameter(requestDate));

public static String mapToUrlParameter(Map<String, String> input){
    StringBuilder url_parameter = new StringBuilder();
    for (Map.Entry<String, String> entry : input.entrySet()) {
        if (url_parameter.length() != 0){
            url_parameter.append("&");
        }
        String key = entry.getKey();
        String value = (String) entry.getValue();
        if (value != null && !value.isEmpty()){
            url_parameter.append(key + "=" + URLEncoder.encode(value));
        }
    }
    return url_parameter.toString();
}
```

Step 4:接收Open ID Auth response code

說明:

完成第三步驟後 redirect_uri會收到code與state(如果有auth有帶此參數才会有)，code 參數是一種**一次性授權碼**，您的伺服器可以交換token和 ID token。您的伺服器會傳送 HTTPS POST 要求來發出這個交換。系統會將 POST 要求傳送至token_endpoint(step 2 可以取得)，state參數可自行與Step2所發起的state參數驗證是否相同，確認response無經過偽造。

接收方式：

```
@GetMapping("/OpenId/CallBack")
public Map<String, Object> CallBack(HttpServletRequest request) {

    String code = request.getParameter("code");
    String state = request.getParameter(Const.STRING_STATE);
```

Step 5:發起Token驗證取得Token

API：

Token

Authentication：none

Method：POST

Service path：[Keyper Server URL]/KeyperManagement/keyper/service/OpenId/token

Service Input：

Content-Type：application/x-www-form-urlencoded

format：parameter

Name	Type	Required	Description
grant_type	string	yes	[refresh_token]or[authorization_code]
code	string	no	if grant_type is 'authorization_code' this is required
refresh_token	string	no	if grant_type is 'refresh_token' this is required
redirect_uri	string	yes	
client_id	string	yes	
client_secret	string	yes	

Service Output：

format：Json

Name	Type	Description
access_token	String	
expires_in	String	
refresh_token	String	

scope	string	
token_type	string	
id_token	String	

Demo Code :

```
String code = request.getParameter("code");
String state = request.getParameter(Const.STRING_STATE);
String clientId = openIdConst.getClientId();
String clientSecret = openIdConst.getClientSecret();
String redirectUri = urlConst.getOpenIdCallBack();
String grantType = "authorization_code";

Map postdata = new HashMap();
postdata.put("grant_type", grantType);
postdata.put("client_id", clientId);
postdata.put("client_secret", clientSecret);
postdata.put("redirect_uri", redirectUri);
postdata.put("code", code);

String postResponse = Methods.doPost(urlConst.getOpenId_token_endpoint(), Methods.mapToUriParameter(postdata));
```

```
public static String doPost(String targetUrl, String postData) throws IOException {

    logger.info("doPost url : " + targetUrl + " , data : " + postData);
    URL url = new URL(targetUrl);

    // Open connection
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();

    // Set request method to POST
    connection.setRequestMethod("POST");

    // Enable input/output streams
    connection.setDoOutput(true);
    connection.setDoInput(true);

    // Set request headers
    connection.setRequestProperty("Content-Type", "application/x-www-form-urlencoded");

    // Write POST data to output stream
    try (DataOutputStream outputStream = new DataOutputStream(connection.getOutputStream())) {
        outputStream.write(postData.getBytes(StandardCharsets.UTF_8));
        outputStream.flush();
    }
}
```

```

// Get response code
int responseCode = connection.getResponseCode();

StringBuilder response = new StringBuilder();
// Read response data
try (BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()))) {
    String line;
    while ((line = reader.readLine()) != null) {
        response.append(line);
    }
}
logger.info(response.toString());

// Close connection
connection.disconnect();
return response.toString();
}

```

```

public static String mapToUrlParameter(Map<String, String> input){
    StringBuilder url_parameter = new StringBuilder();
    for (Map.Entry<String, String> entry : input.entrySet()) {
        if (url_parameter.length() != 0){
            url_parameter.append("&");
        }
        String key = entry.getKey();
        String value = (String) entry.getValue();
        if (value != null && !value.isEmpty()){
            url_parameter.append(key + "=" + URLEncoder.encode(value));
        }
    }
    return url_parameter.toString();
}

```

```
{
  "access_token": "9ba047aac9cc9eb4c9157ce3da9ad6db2af46a5c37e22cb1799bf807aa934810b585a3232adcd49dbd96b81d2b95ae58f32bb5ea302c566bfc0cbe3a86486cb3f02cceb55c8ca7e105abc3e57efbd1365aae32ff7804d97f3ffe3d617c7d56ad77584f5a48296601e502c202c6e9a3ed09d5991d53348bf8ddf23a4d2ad1bfe525e03f2ed92ce7b65caf5efa485087a6",
  "refresh_token": "7ad67c90eeb0d95d060c6a021fe9458b8891c4028a48ed83919497ce11e4cec9c55aec0b2f72e7574620943d9021952ac5261fec324ee7b6dff7228a0f0c19877205a4c08eb0b0b909b2fc8560995a1d78ebe9586161b69a5091f5eca86ea76bd22050ffa3e3c4d343a1a7d68241fe8260ca287eaf27e5b9ed59f81682a3d17525fd70cfac340d305c27974435ed76dc2af7680206de6601d92a6df0f831ecfe4f6eaca06dcfa30f977d09c90a87e24afe16a566cc39a0892ac6141722d1b4a33409338660e23e6eb73cabdf97a6bb04bc48102aecc229a934755c3e52e96460e2bb923cd502dbab966a568c8d6ddd97a7ce6f4d7ab1179126a3856ed7d80dc6ff502d9cc772e5ec37502c83789",
  "scope": "openid profile",
  "id_token": "eyJ0eXAiOiJKV1QiLCJraWQiOiJmeEpYYiJYdWdGQVZRzaEdUSEhtakR6bVNGUXhnaFhvdmtiVWJkRnliIiwiaWYwbnIjoIUMyNTYifQ.eyJpc3MiOiJodHRwczovL2tleXB1cmRldi5rZXI4ZW50aWMuY29tOjg0NDMvIiwic3ViOiJoiMTE1NSIsImF1ZCI6ImQ4M2JmOWQ2LTgyOTQtNGMzOC04NWZILTViMTQzNzNlM2Y2MSIsImV4cCI6MTcyMzEwMzY4MCwiaWF0IjoxNzIzMTA3MDgwLCJub25jZSI6ImFmMTNkY2JhLWFmMDQtNGZlMjY0ZWJkLTU5YThkYmFmMzdkMSIsIm5hbWUiOiJFZGRpZVhpZSJ9.PSRLLoR6WgE7-bUyDRqqG6RIimbL_uKZZW-vxW73r86iRvnekycYay5WLlnoOy4nmsww2KhkNacPoRZOSARl6t4ofG6jh6Ufc4os4wheYfsTJHRNvi_bG0ghDS14vR_ZON9FogIONREwU_5c39acMxUBLV_spgqeltpdW59luVOVok0kur5gikUHpTmPU6WKPNhCcoywADnu0w2AHeIhsJWGEhhT5IE_IVVvQJf8RiK8QUuhEINQIr5WDCfgR2-UeioeNcyHf5kMGJZUH34nkUoUSTEBuB9JPoJvvlAhJ_HuFo2w8PPZSko0Pv_ecV473woPRR3kTEKpUyl_rVJ6KLyg",
  "token_type": "Bearer",
  "expires_in": "3599"
}
```

將Step 5取得的postResponse String 轉成 Json格式，取得id_token。(範例中用Gson)

驗JWT 時間是否過期

```
SignedJWT signedJWT = SignedJWT.parse(idToken);
Date expirationTime = signedJWT.getJWTClaimsSet().getExpirationTime();
if (expirationTime.before(new Date())) {
    response.clear();
    response.put("error", "idToken is expired");
    logger.info(transactionInvoice, gson.toJson(response));
    return response;
}
```

取得Step2 的jwks_uri ， 取得JWT key 並驗簽

```
JWKSet jwkSet = JWKSet.load(new URL(urlConst.getHeader().getJwks_uri()));
String kid = signedJWT.getHeader().getKeyID();
RSAKey rsaKey = null;
if (kid != null){
    rsaKey = (RSAKey) jwkSet.getKeyByKeyId(kid);
}else {
    rsaKey = (RSAKey)jwkSet.getKeys().get(0);
}
RSAPublicKey publicKey = rsaKey.toRSAPublicKey();
// 验证签名
JWSVerifier verifier = new RSASSAVerifier(publicKey);
if (!signedJWT.verify(verifier)){
    response.clear();
    response.put("error", "idToken verify fail");
    logger.info(transactionInvoice, gson.toJson(response));
    return response;
}
```

取得UserName

```
String name = reponseMap.get("name").toString();
```

JWT内容Example

Encoded	PASTE A TOKEN HERE	Decoded	EDIT THE PAYLOAD AND SECRET
	<pre>eyJ0eXAiOiJKV1QiLCJraWQoIjMeEpYYlJYdWdGQQVzaEdUSEhtakR6bVNGUXhnaFhdmtiVWJkRnliIiwiaWF0eSI6MTYyNTYifQ.eyJpc3MiOiJodHRwczovL2tleXBldmRldi5rZXI4ZW50aWMuY29tOjg0NDMvIiwic3ViOiJTE1NSIsImF1ZCI6ImQ4M2JmOWQzLTgyOTQtNGMzOC04NWZlLVltMQZnNlM2Y2MSIsImV4cCI6MTcyMzEwMzY0M2cwiaWF0IjoxNzIxMTAwMDgwLClub25jZS16ImFmMTNKY2JhLWFMMDQtNGZlMy04ZWJkLTU5THkybmFlMzdKMSIsIm5hbWUiOiJFZFGRZVhpZSJ9.PSRLLoR6WgE7-bUyDRqG6RIhmbl_uKZZW-vxW73r86iRvnekycYay5WLlnoOy4nmsww2KhkNAcPoRZOSAR16t4ofG6jh6Ufc4os4wheYfsTJHRNvi_bG0ghDS14vr_ZON9Fog1ONREwU_5c39acMxUBLV_spqeg1tpdW591uVOvk0kur5gikUHPTmpU6WKPhnCcoywADnu0w2AHeIhsJWGEhhT5IE_1VVvQJs8RiK8QUuhE1NQIR5WDCfgr2-UeioeNcyHf5kgMGJZUH34nkUoUSTEuB9JPoJvv1AhJ_HuFo2w8PPZSk0Pv_ecV473woPRR3kTEKpUyl_rVJ6KLpy</pre>		<pre>Header: Algorithm & Token Type { "typ": "JWT", "kid": "fxJXbRXugFATshGTHmjDzmSFQxghXovkbUbdFyb", "alg": "RS256" } Payload: Data { "sub": "1155", "aud": "d83bf9d6-8294-4c38-85fe-5b14373ef61", "exp": 1723103600, "iat": 1723100000, "nonce": "af13dcba-af04-4fe3-8ebd-69a8dbae37d1", "name": "EddieXie" } Verify Signature RSASHA256(base64UrlEncode(header) + ".", + base64UrlEncode(payload), Public key in SPKI, PKCS #1, X.509 Certificate, or JWK string format.)</pre>

Step 7：token過期 可使用 Refresh Token 取得新的token

API：

Token

Authentication：none

Method：POST

Service path：[Keyper Server URL]/KeyperManagement/keyper/service/OpenId/token

Service Input：

Content-Type：application/x-www-form-urlencoded

format：parameter

Name	Type	Required	Description
grant_type	string	yes	[refresh_token]or[authorization_code]
code	string	no	if grant_type is 'authorization_code' this is required
refresh_token	string	no	if grant_type is 'refresh_token' this is required
redirect_uri	string	yes	
client_id	string	yes	
client_secret	string	yes	

Service Output：

format：Json

Name	Type	Description
access_token	String	
expires_in	String	
refresh_token	String	
scope	string	
token_type	string	
id_token	String	

Demo Code :

```
String clientId = openIdConst.getClientId();
String clientSecret = openIdConst.getClientSecret();
String redirectUri = urlConst.getOpenIdCallBack();
String grantType = "refresh_token";

Map postdata = new HashMap();
postdata.put("grant_type", grantType);
postdata.put("client_id", clientId);
postdata.put("client_secret", clientSecret);
postdata.put("redirect_uri", redirectUri);
postdata.put("refresh_token", "Your refreshToken");

String postResponse = Methods.doPost(urlConst.getOpenId_token_endpoint(), Methods.mapToUrlParameter(postdata));
```

response Example

```
{
  "access_token": "193bfe329edccff32053693a0bf995e3b10e6148d773691e603483df274653288f28ae6c46e5dfa07373f220d94bdc
cd2345ef2f97c369155e0a5d09076c5ccea915f5330a2d81ccee2a785e725f5d97a96a4cd5139c02a49c451152bb8428f80264cf75ca31f
350c6e830afc79e099b9e94a318c56fc97941251f8d387903f8ca19962f65be1855b174bd21b8315f0",
  "scope": "openid profile",
  "id_token": "eyJ0eXAiOiJKV1QiLCJraWQiOiJmeEpYYlJYdWdGQVRzaEdUSEhtakR6bVNGUXhnaFhvdmtiVWJkRnliIiwiaY
WxnIjoiaUIMyNTYifQ.eyJpc3MiOiJodHRwczovL2tleXBldmRldi5rZXI4ZW50aWMuY29tOjg0NDMvIiwic3ViIjoiaUIMTE1NSIsImF1
ZCI6ImQ4M2JmOWQ2LTgyOTQtNGMzOC04NWZlTViMTQzNzNlM2Y2MSIsImV4cCI6MTcyMzE3ODEzNywiaWF0IjoxNz
IzMTc0NTM3LzJub25jZSI6IjBhNWNmYzVhLWE4ODItNGEzMy05ZWwLTKwNjE5ZjA5Y2ZlYSIsIm5hbWUiOiJFZGRpZV
hpZSJ9.QN3LshZ79AJJBsuUpyHkJR4gJwvzgOYDSyHu4-EbJuOWXx97znYu83L7EL5UprOr0z761ez81Y-aScoTak2Ce-cuckMI
BSOgvg1zCEYhk8mcSYhsbzOGRdfVSfODRj5t27qANSwj7kWaU6G1VerMQvN4YhuIKhpQC4mi8wkYtQOTJy9YCEZuNk1Olc
sRK3vs789j8neNw_pTXA6VsoHRNhE2VoMk9IqMAV_ScSyzlIOB43hoIIS0JldM97h-0DshaQ1uLEwJ46hThkvxbFGKTPyeoe0d
hrogFdIsjBR7NM8i51NivpRGJ8A3H8RXHAeu3v-nH7LqvGMhgbW9xdpA",
  "token_type": "Bearer",
  "expires_in": "3599"
}
```

後續驗證 JWT方式 及取得 user 資訊方法與Step6一樣。

備註：

引入maven套件

```
<dependency>
  <groupId>com.nimbusds</groupId>
  <artifactId>nimbus-jose-jwt</artifactId>
  <version>9.37.3</version>
</dependency>
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
</dependency>
<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-api</artifactId>
  <version>0.11.2</version>
</dependency>

<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-impl</artifactId>
  <version>0.11.2</version>
</dependency>

<dependency>
  <groupId>io.jsonwebtoken</groupId>
  <artifactId>jjwt-jackson</artifactId>
  <version>0.11.2</version>
</dependency>
<dependency>
  <groupId>org.json</groupId>
  <artifactId>json</artifactId>
  <version>20180130</version>
</dependency>
```