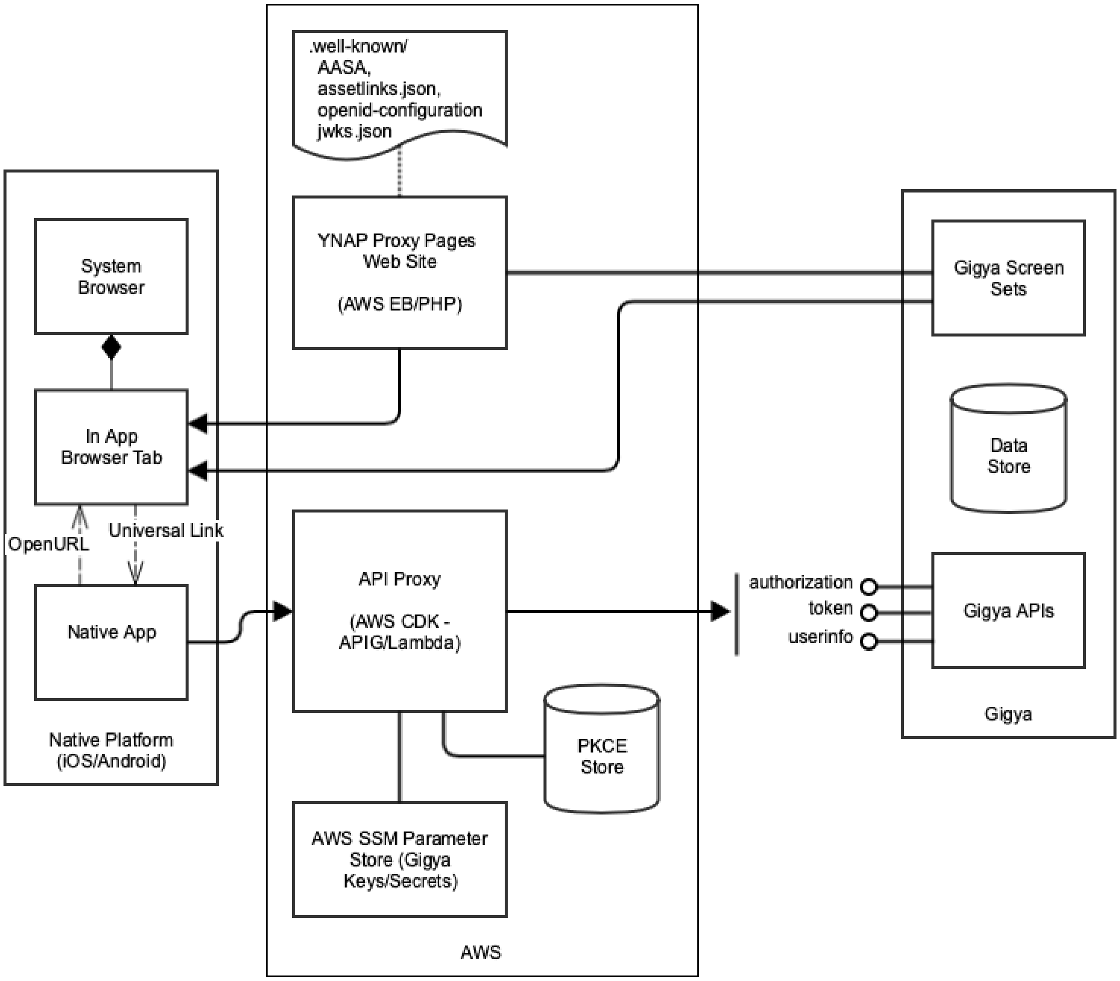
Native App OIDC with Gigya

Introduction

This page describes the work that was done to prove that Gigya can be configured as an Open ID Connect Provider and made to work with a standards compliant native client.

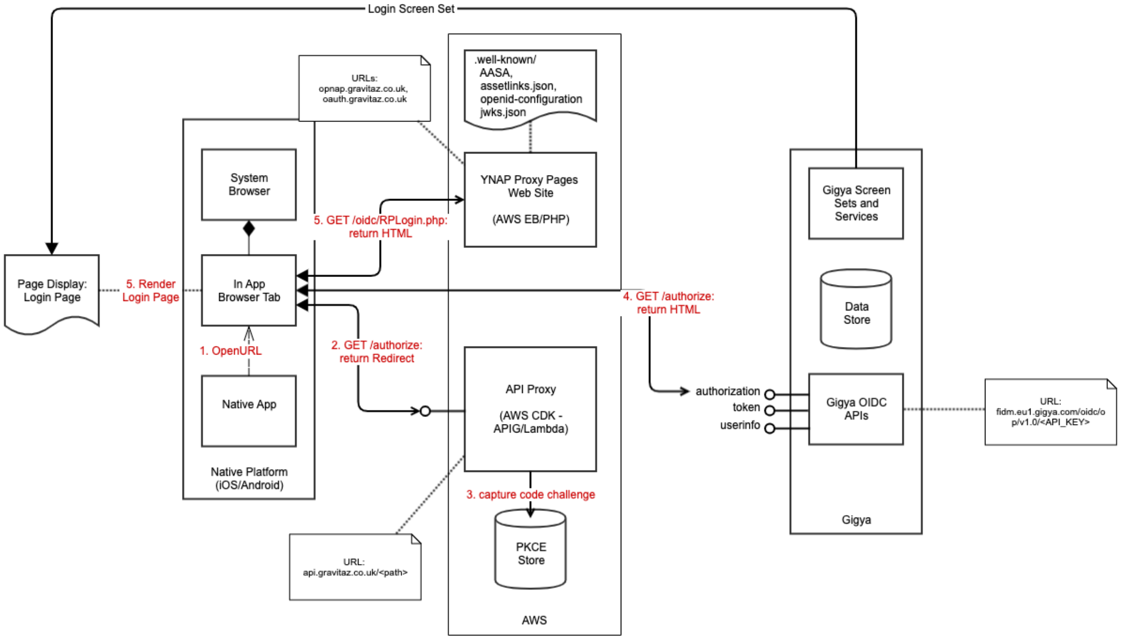
The client used for this exercise is http://appauth.io

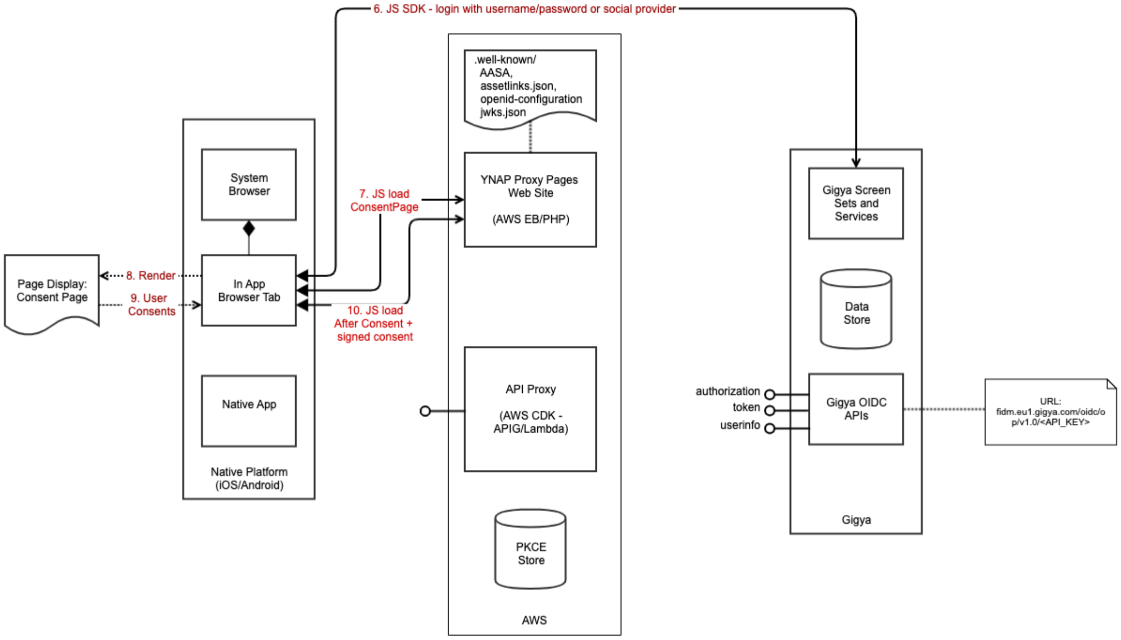
Since OIDC is a secure protocol, it relies upon non-self signed SSL certificates for https communication. For the purpose of this exercise I have used my own personal domain, and provisioned SSL certificates using AWS.

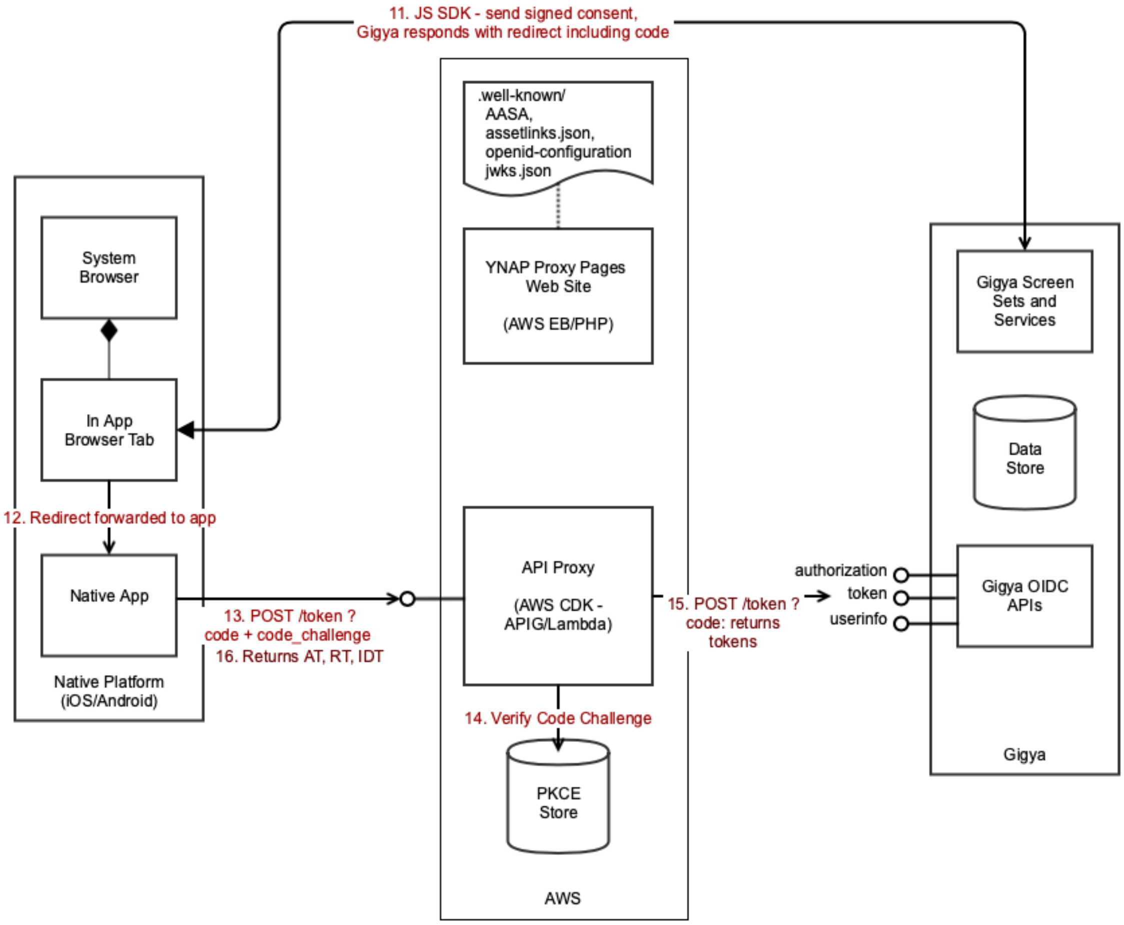


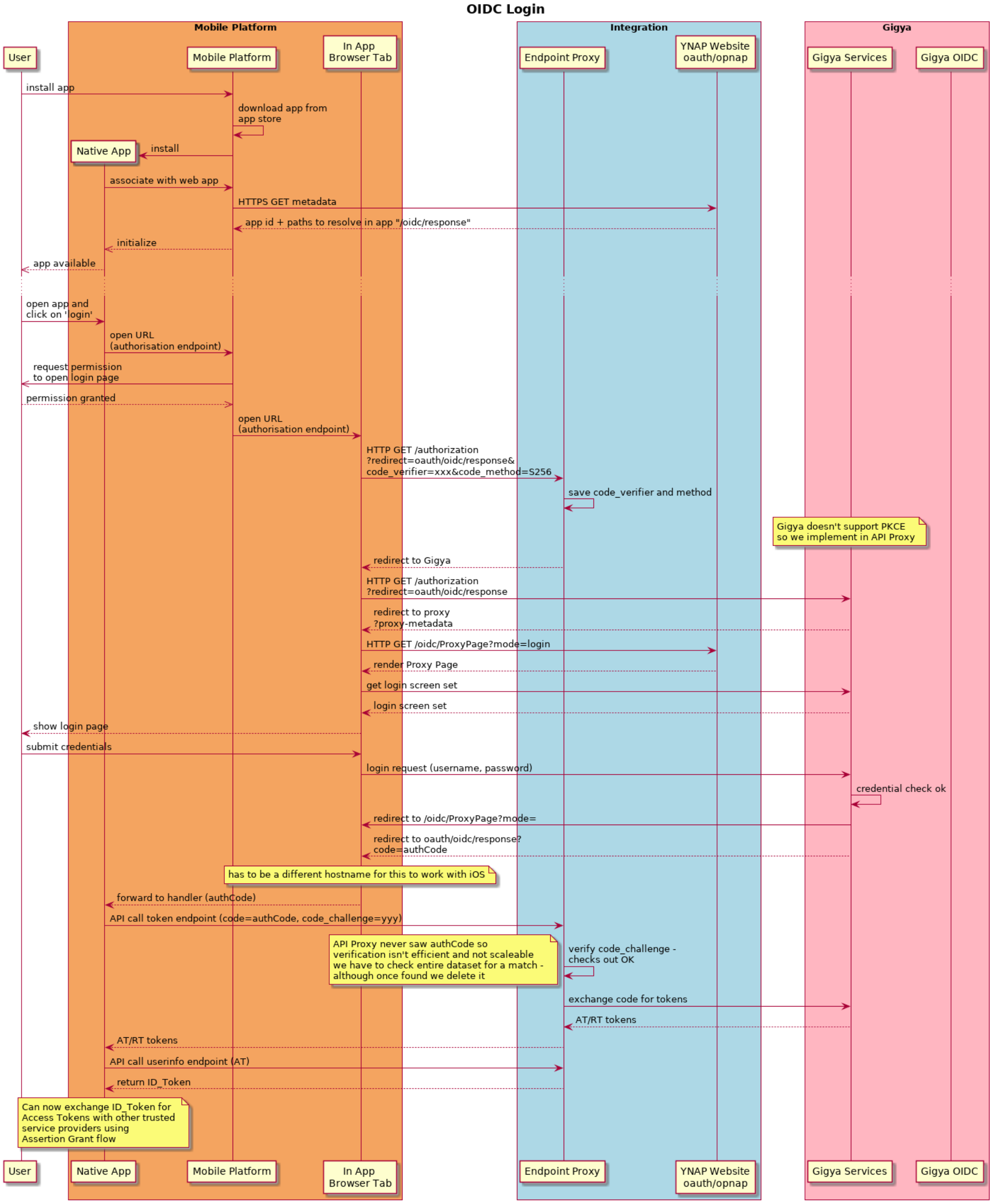
Implementation

Login


Login Screen Set
URLs:
opnap.gravitaz.co.uk,
oauth.gravitaz.co.uk
.well-known/
AASA,
assetlinks.json,
openid-configuration
jwks.json
YNAP Proxy Pages
Web Site
(AWS EB/PHP)
System
Browser
5. GET /oidc/RPLogin.php:
return HTML
Page Display:
Login Page
5. Render
Login Page
In App
Browser Tab
4. GET /authorize:
return HTML
Gigya Screen
Sets and
Services
Data
Store
1. OpenURL
2. GET /authorize:
return Redirect
API Proxy
(AWS CDK -
APIG/Lambda)
authorization
token
userinfo
Gigya OIDC
APIs
URL:
fidm.eu1.gigya.com/oidc/o
p/v1.0/<API_KEY>
Native App
Native Platform
(iOS/Android)
3. capture code challenge
PKCE
Store
Gigya
URL:
api.gravitaz.co.uk/<path>
AWS


6. JS SDK - login with username/password or social provider
.well-known/
AASA,
assetlinks.json,
openid-configuration
jwks.json
System
Browser
7. JS load
ConsentPage
YNAP Proxy Pages
Web Site
(AWS EB/PHP)
Gigya Screen
Sets and
Services
Page Display:
Consent Page
8. Render
9. User
Consents
In App
Browser Tab
10. JS load
After Consent +
signed consent
Data
Store
API Proxy
(AWS CDK -
APIG/Lambda)
authorization
token
userinfo
Gigya OIDC
APIs
URL:
fidm.eu1.gigya.com/oidc/o
p/v1.0/<API_KEY>
Native App
Native Platform
(iOS/Android)
Gigya
PKCE
Store
AWS


11. JS SDK - send signed consent,
Gigya responds with redirect including code
.well-known/
AASA,
assetlinks.json,
openid-configuration
jwks.json
System Browser
In App Browser Tab
12. Redirect forwarded to app
Native App
Native Platform (iOS/Android)
13. POST /token ?
code + code_challenge
16. Returns AT, RT, IDT
YNAP Proxy Pages Web Site
(AWS EB/PHP)
API Proxy
(AWS CDK - APIG/Lambda)
14. Verify Code Challenge
PKCE Store
AWS
15. POST /token ?
code: returns
tokens
authorization
token
userinfo
Gigya Screen Sets and Services
Data Store
Gigya OIDC APIs
Gigya

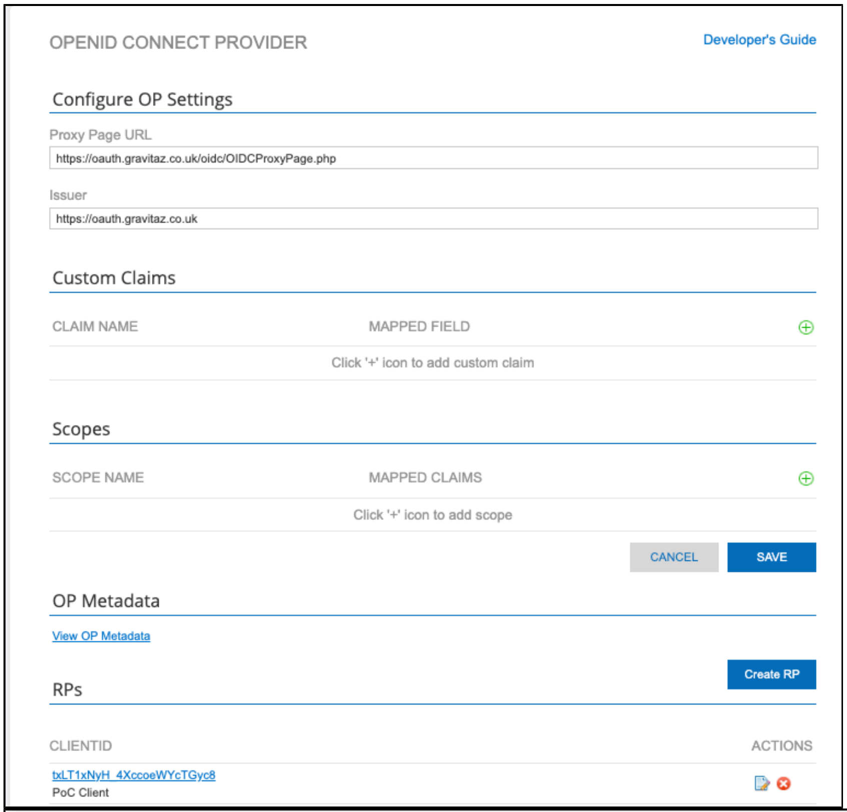

OIDC Login
Mobile Platform
Integration
Gigya
User
Mobile Platform
In App Browser Tab
Endpoint Proxy
YNAP Website oauth/opnap
Gigya Services
Gigya OIDC
install app
download app from app store
Native App
install
associate with web app
HTTPS GET metadata
app id + paths to resolve in app "/oidc/response"
initialize
app available
open app and click on 'login'
open URL (authorisation endpoint)
request permission to open login page
permission granted
open URL (authorisation endpoint)
HTTP GET /authorization ?redirect=oauth/oidc/response& code_verifier=xxx&code_method=S256
save code_verifier and method
Gigya doesn't support PKCE so we implement in API Proxy
redirect to Gigya
HTTP GET /authorization ?redirect=oauth/oidc/response
redirect to proxy ?proxy-metadata
HTTP GET /oidc/ProxyPage?mode=login
render Proxy Page
get login screen set
login screen set
show login page
submit credentials
login request (username, password)
credential check ok
redirect to /oidc/ProxyPage?mode=
redirect to oauth/oidc/response? code=authCode
has to be a different hostname for this to work with iOS
forward to handler (authCode)
API call token endpoint (code=authCode, code_challenge=yyy)
API Proxy never saw authCode so verification isn't efficient and not scaleable we have to check entire dataset for a match - although once found we delete it
verify code_challenge - checks out OK
exchange code for tokens
AT/RT tokens
AT/RT tokens
API call userinfo endpoint (AT)
return ID_Token
Can now exchange ID_Token for Access Tokens with other trusted service providers using Assertion Grant flow
User
Native App
Mobile Platform
In App Browser Tab
Endpoint Proxy
YNAP Website oauth/opnap
Gigya Services
Gigya OIDC

Configuration

Back End

Gigya Configuration

Gigya can be configured for Open ID Connect, but support for the protocol is incomplete.



OPENID CONNECT PROVIDER

Configure OP Settings

Proxy Page URL

https://oauth.gravitaz.co.uk/oidc/OIDCProxyPage.php

Issuer

https://oauth.gravitaz.co.uk

Custom Claims

CLAIM NAME	MAPPED FIELD	⊕
Click '+' icon to add custom claim		

Scopes

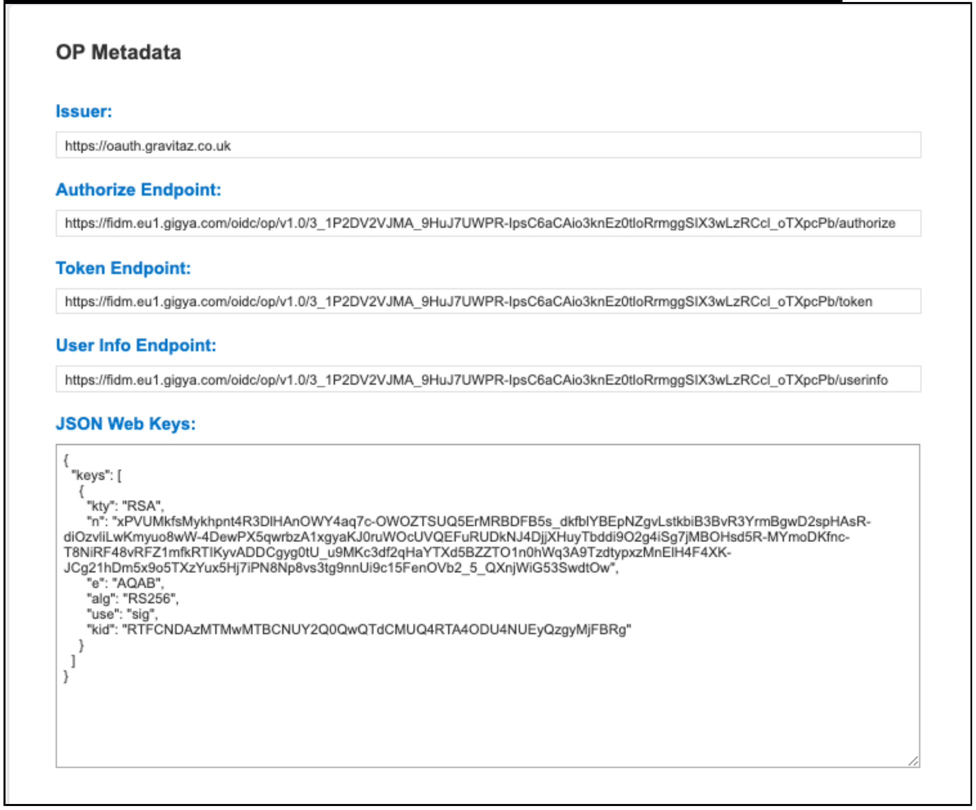
SCOPE NAME	MAPPED CLAIMS	⊕
Click '+' icon to add scope		

OP Metadata

View OP Metadata

RPs

CLIENTID	ACTIONS
txLT1xNyH_4XccoeWYcTGyc8 PoC Client	



OP Metadata

Issuer:

https://oauth.gravitaz.co.uk

Authorize Endpoint:

https://fidm.eu1.gigya.com/oidc/op/v1.0/3_1P2DV2VJMA_9HuJ7UWPR-IpsC6aCAio3knEz0tloRrmggSIX3wLzRCcl_oTXpcPb/authorize

Token Endpoint:

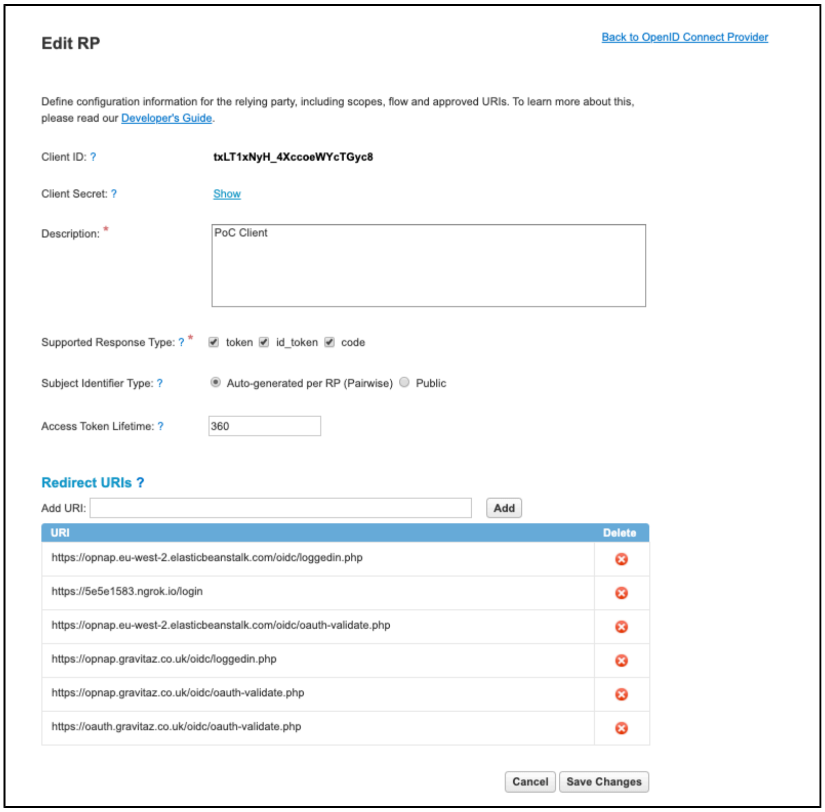
https://fidm.eu1.gigya.com/oidc/op/v1.0/3_1P2DV2VJMA_9HuJ7UWPR-IpsC6aCAio3knEz0tloRrmggSIX3wLzRCcl_oTXpcPb/token

User Info Endpoint:

https://fidm.eu1.gigya.com/oidc/op/v1.0/3_1P2DV2VJMA_9HuJ7UWPR-IpsC6aCAio3knEz0tloRrmggSIX3wLzRCcl_oTXpcPb/userinfo

JSON Web Keys:

```
{
  "keys": [
    {
      "kty": "RSA",
      "n": "xPVUMkfsMykhpnt4R3DIHAnOWY4aq7c-OWOZTSUQ5ErMRBDFB5s_dkfblYBEpNZgvLstkbiB3BvR3YrmBgwD2spHAsR-
diOzvliLwKmyuo8wW-4DewPX5qwrbzA1xgyaKJ0ruWOcUVQEFuRUDkNJ4DjjXHuyTbddi9O2g4iSg7jMBOHsd5R-MYmoDKfnc-
T8NiRF48vRFZ1mfkRTIKyvADDCgyg0tU_u9MKc3df2qHaYTXd5BZZTO1n0hWq3A9TzdtypxzMnElH4F4XK-
JCg21hDm5x9o5TXzYux5Hj7iPN8Np8vs3tg9nnUi9c15FenOVb2_5_QXnjWiG53SwdtOw",
      "e": "AQAB",
      "alg": "RS256",
      "use": "sig",
      "kid": "RTFCNDAzMTMwMTBCNUY2Q0QwQTdCMUQ4RTA4ODU4NUEyQzgyMjFBRg"
    }
  ]
}
```



Edit RP

Define configuration information for the relying party, including scopes, flow and approved URIs. To learn more about this, please read our Developer's Guide.

Client ID: ? **txLT1xNyH_4XccoeWYcTGyc8**

Client Secret: ? Show

Description: * PoC Client

Supported Response Type: ? * ☑ token ☑ id_token ☑ code

Subject Identifier Type: ? ● Auto-generated per RP (Pairwise) ○ Public

Access Token Lifetime: ? 360

Redirect URIs ?

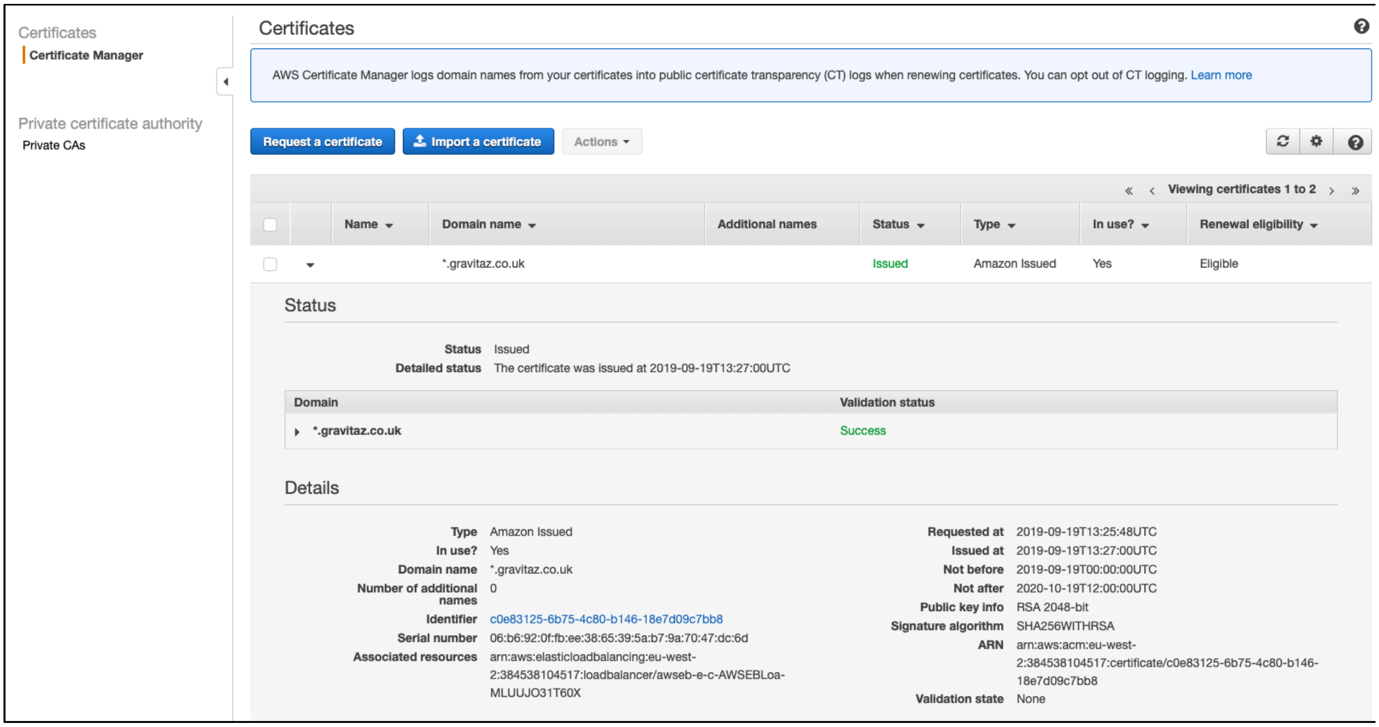
Add URI:

URI	Delete
https://opnap.eu-west-2.elasticbeanstalk.com/oidc/loggedin.php	☒
https://5e5e1583.ngrok.io/login	☒
https://opnap.eu-west-2.elasticbeanstalk.com/oidc/oauth-validate.php	☒
https://opnap.gravitaz.co.uk/oidc/loggedin.php	☒
https://opnap.gravitaz.co.uk/oidc/oauth-validate.php	☒
https://oauth.gravitaz.co.uk/oidc/oauth-validate.php	☒

Proxy Page (PHP on AWS Elastic Beanstalk)

Gigya doesn't provide a standalone web application that supports login, consent flows. Instead we need to create a Proxy service as described here.
A PHP website has been created and deployed to AWS Elastic Beanstalk here: here

So this will work with native clients Elastic Beanstalk was mapped to two subdomains of a a personal domain with an AWS SSL certificate...



Certificates

AWS Certificate Manager logs domain names from your certificates into public certificate transparency (CT) logs when renewing certificates. You can opt out of CT logging. Learn more

« ‹ Viewing certificates 1 to 2 › »

□		Name	Domain name	Additional names	Status	Type	In use?	Renewal eligibility
□	▾		*.gravitaz.co.uk		Issued	Amazon Issued	Yes	Eligible

Status

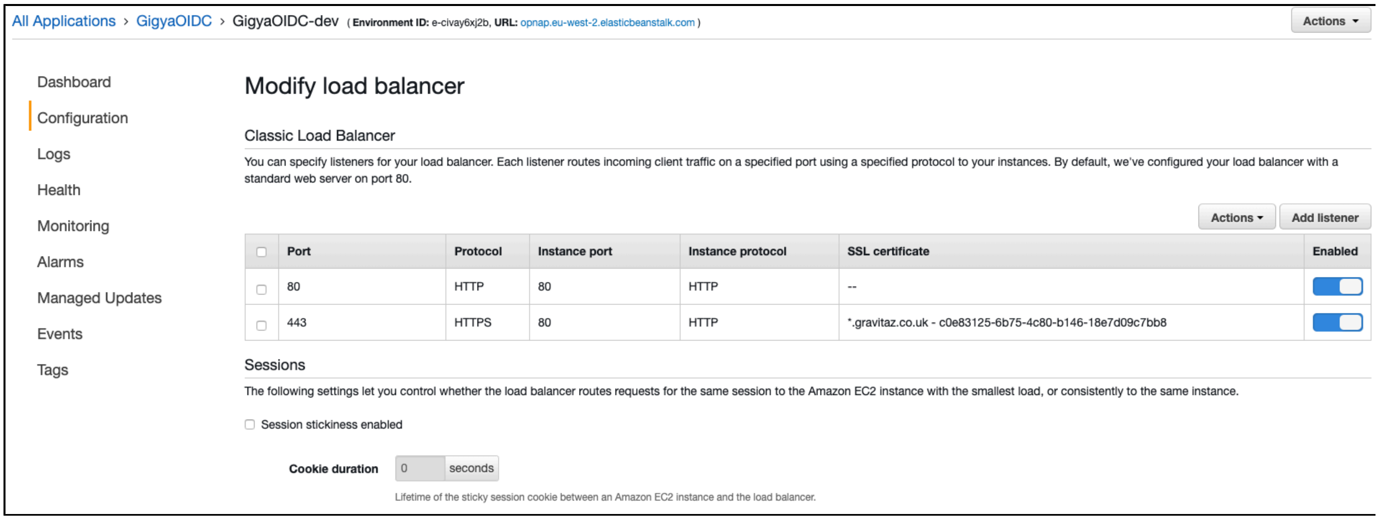
Status Issued
Detailed status The certificate was issued at 2019-09-19T13:27:00UTC

Domain	Validation status
▸ *.gravitaz.co.uk	Success

Details

Type	Amazon Issued	Requested at	2019-09-19T13:25:48UTC
In use?	Yes	Issued at	2019-09-19T13:27:00UTC
Domain name	*.gravitaz.co.uk	Not before	2019-09-19T00:00:00UTC
Number of additional names	0	Not after	2020-10-19T12:00:00UTC
		Public key info	RSA 2048-bit
Identifier	c0e83125-6b75-4c80-b146-18e7d09c7bb8	Signature algorithm	SHA256WITHRSA
Serial number	06:b6:92:0f:fb:ee:38:65:39:5a:b7:9a:70:47:dc:6d	ARN	arn:aws:acm:eu-west-2:384538104517:certificate/c0e83125-6b75-4c80-b146-18e7d09c7bb8
Associated resources	arn:aws:elasticloadbalancing:eu-west-2:384538104517:loadbalancer/awseb-e-c-AWSEBLoa-MLUUJO31T60X	Validation state	None

The certificate was associated with the load balancer of an Elastic Beanstalk deployment with subdomain 'opnap':



(**Environment ID:** e-civay6xj2b, **URL:** opnap.eu-west-2.elasticbeanstalk.com)

Modify load balancer

Classic Load Balancer

You can specify listeners for your load balancer. Each listener routes incoming client traffic on a specified port using a specified protocol to your instances. By default, we've configured your load balancer with a standard web server on port 80.

□	Port	Protocol	Instance port	Instance protocol	SSL certificate	Enabled
□	80	HTTP	80	HTTP	--	●
□	443	HTTPS	80	HTTP	*.gravitaz.co.uk - c0e83125-6b75-4c80-b146-18e7d09c7bb8	●

Sessions

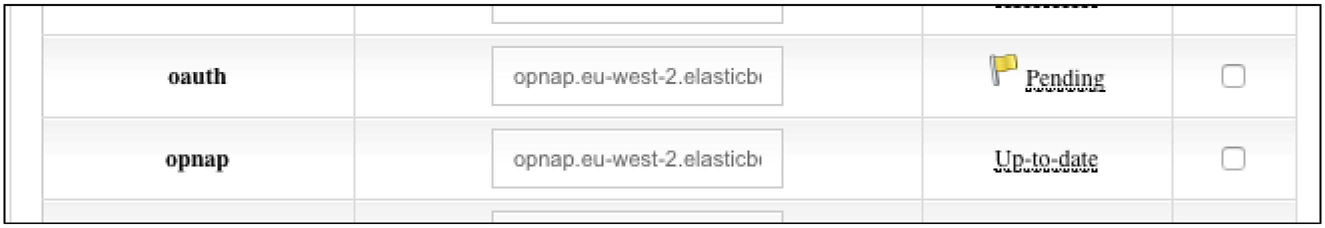
The following settings let you control whether the load balancer routes requests for the same session to the Amazon EC2 instance with the smallest load, or consistently to the same instance.

□ Session stickiness enabled

Cookie duration 0 seconds

Lifetime of the sticky session cookie between an Amazon EC2 instance and the load balancer.

Two separate subdomains were created within the domain 'gravitaz.co.uk' and mapped as CNAMEs to the Elastic Beanstalk deployment.



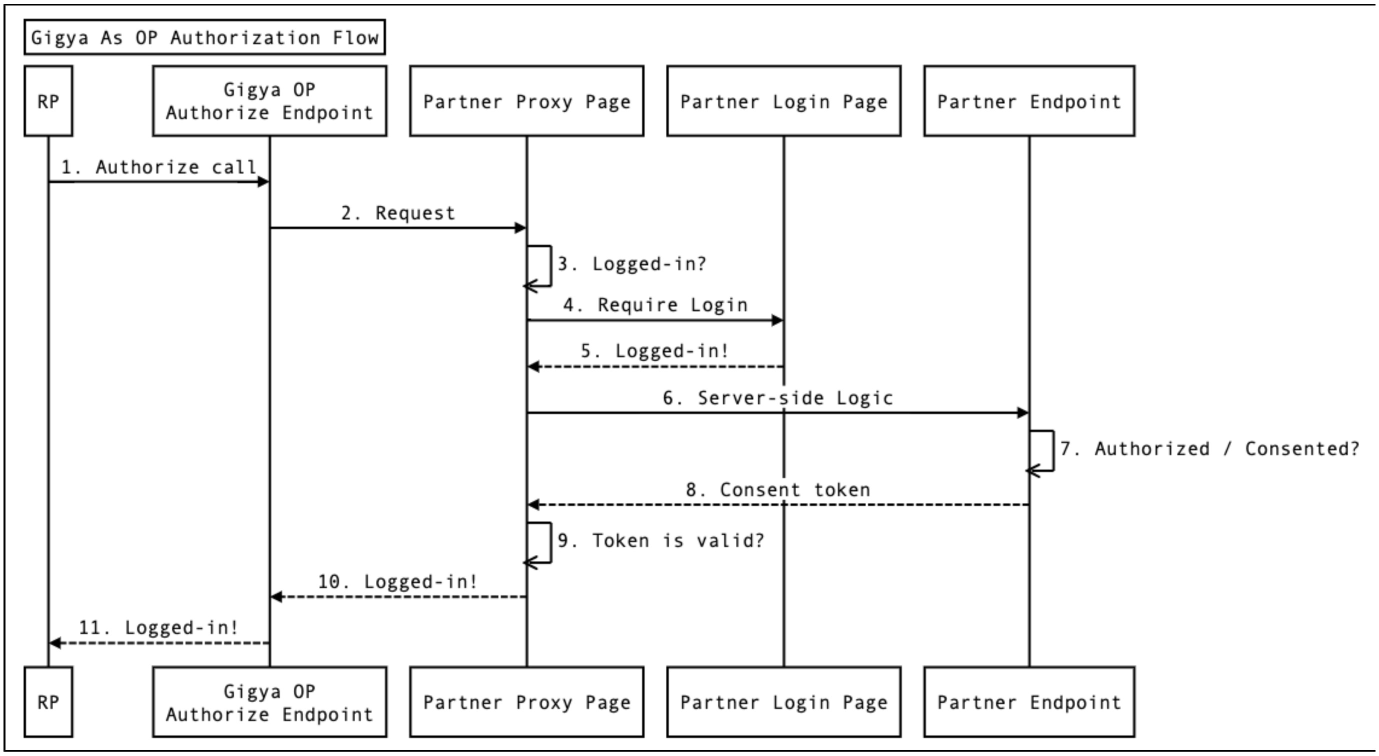
oauth	opnap.eu-west-2.elasticb	Pending	□
opnap	opnap.eu-west-2.elasticb	Up-to-date	□

This resulted in two correctly certified SSL endpoints for the Elastic Beanstalk deployment:

1. https://oauth.gravitaz.co.uk
2. https://opnap.gravitaz.co.uk

Both pointing to https://opnap.eu-west-2.elasticbeanstalk.com

The Elastic Beanstalk deployment was created using PHP software provided by Gigya and available here: https://git.yoox.net/users/ti.holmes/repos/gigyaoidcproxy



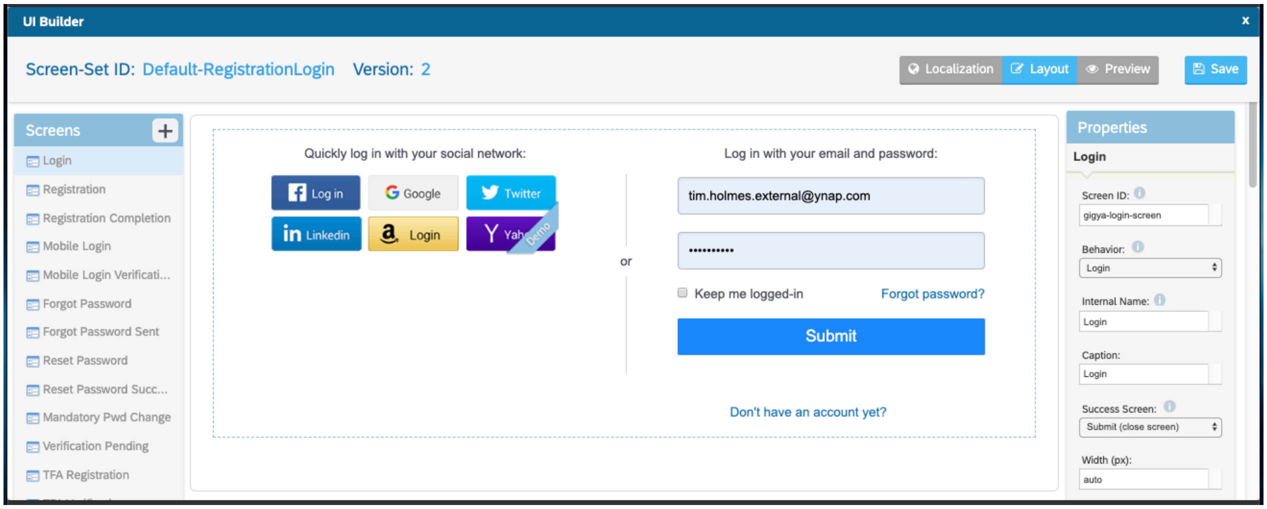
In the above sequence diagram, RP is the Client e.g. iOS or Android App.

Gigya OP Authorize Endpoint is shown in the OP Metadata screenshot.

Partner Pages are provided within the Elastic Beanstalk deployment, at URLs with the path prefix https://oauth.gravitaz.co.uk/oidc/

Partner Proxy Page is OIDCProxyPage.php, which is uses a Gigya supplied javascript library to present the OIDCLoginPage.php, OIDCConsentPage.php and/or OIDCErrorPage.php as required.

OIDCLoginPage.php uses javascript libraries from Gigya to render a Gigya ScreenSet 'RegistrationLogin' in a native browser in-app tab. This can be edited within the Gigya console:



Screen-Set ID: Default-RegistrationLogin Version: 2

tim.holmes.external@ynap.com

••••••••••

☐ Keep me logged-in

Login

Screen ID: gigya-login-screen

Behavior: Login

Internal Name: Login

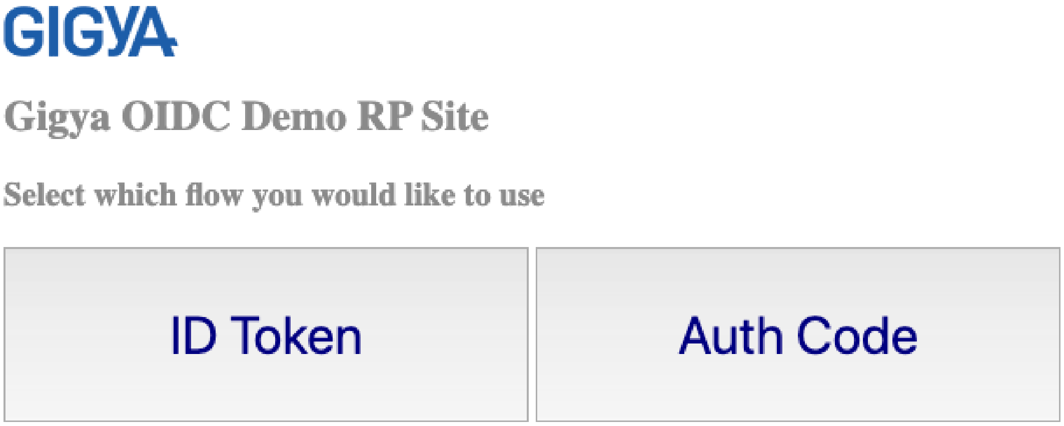
Caption: Login

Success Screen: Submit (close screen)

Width (px): auto

The actual flow is shown below (using a web browser). Note that at step 11 in the diagram above the communicated via a 302 redirect using a pre-configured redirect URL as shown above in the RP Gigya metadata.

1. Open RPLogin.php and click on Auth Code

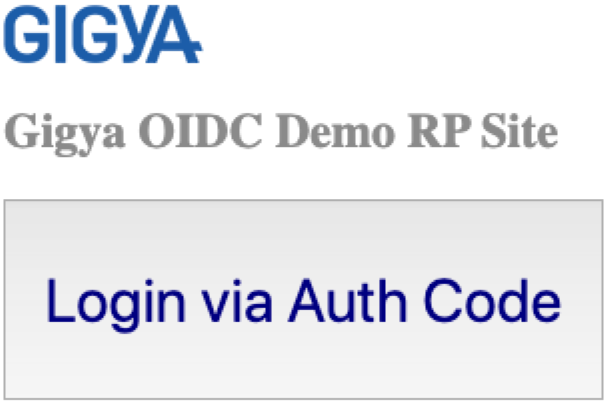


GIGYA

Gigya OIDC Demo RP Site

Select which flow you would like to use

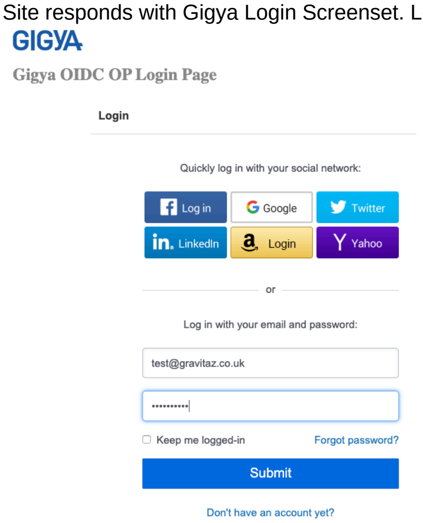
2. Select option 'Login via Auth Code'



GIGYA

Gigya OIDC Demo RP Site

3. Site responds with Gigya Login Screenset. Login with a valid username/password



GIGYA

Gigya OIDC OP Login Page

Login

Quickly log in with your social network:

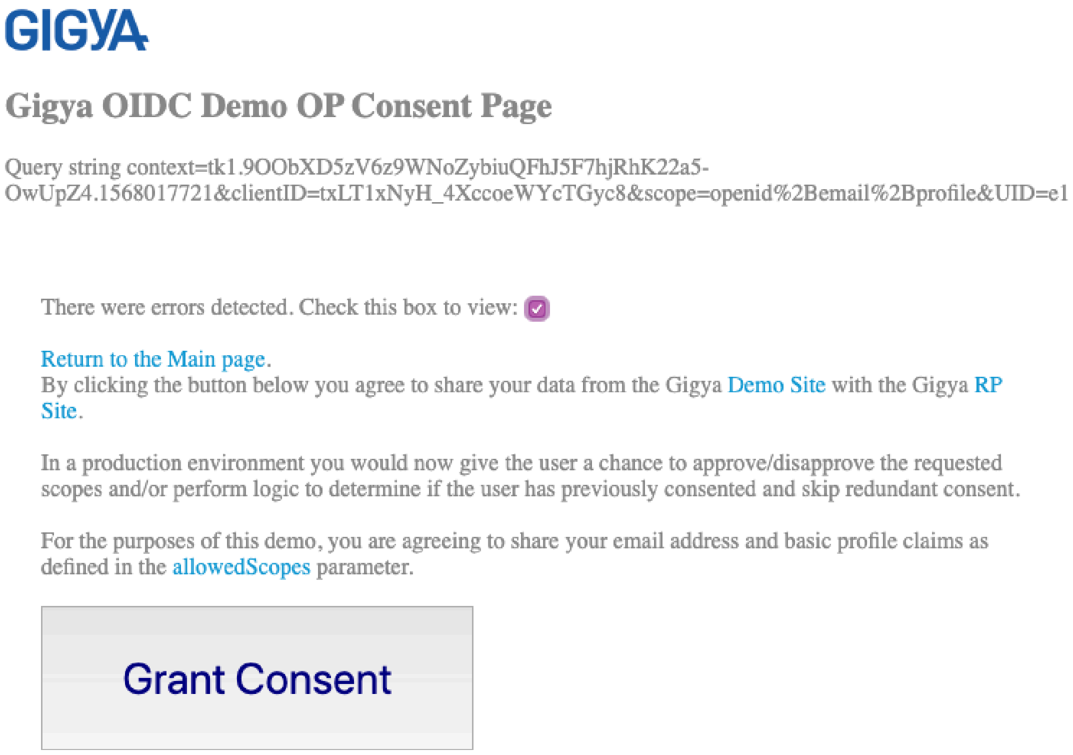
Log in with your email and password:

test@gravitaz.co.uk

••••••••••

□ Keep me logged-in

4. Site responds with a Gigya Consent Page



GIGYA

Gigya OIDC Demo OP Consent Page

Query string context=tk1.9OObXD5zV6z9WNoZybiuQFhJ5F7hjRhK22a5-OwUpZ4.1568017721&clientID=txLT1xNyH_4XccoeWYcTGyc8&scope=openid%2Bemail%2Bprofile&UID=e1

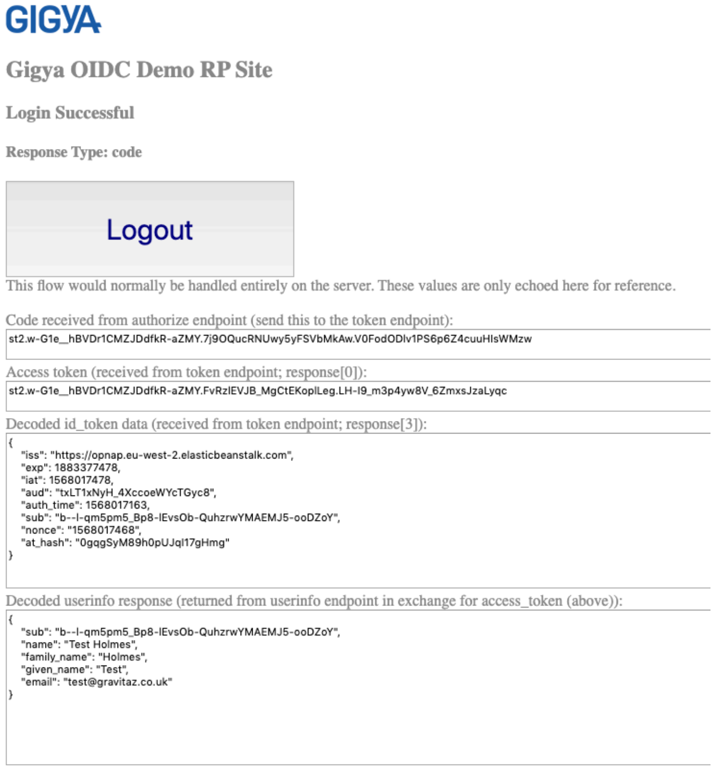
There were errors detected. Check this box to view: ☑

Return to the Main page.
By clicking the button below you agree to share your data from the Gigya Demo Site with the Gigya RP Site.

In a production environment you would now give the user a chance to approve/disapprove the requested scopes and/or perform logic to determine if the user has previously consented and skip redundant consent.

For the purposes of this demo, you are agreeing to share your email address and basic profile claims as defined in the allowedScopes parameter.

5. Click on 'Grant Consent' - page displays the Authorisation Code generated by previous step, then goes on to exchange code for an Access Token and Identity Token.



GIGYA

Gigya OIDC Demo RP Site

Login Successful

Response Type: code

This flow would normally be handled entirely on the server. These values are only echoed here for reference.

Code received from authorize endpoint (send this to the token endpoint):

```
st2.w-G1e__hBVDr1CMZJDdfkR-aZMY.7j9OQucRNUwy5yFSVbMkAw.V0FodODlv1PS6p6Z4cuuHlsWMzw
```

Access token (received from token endpoint; response[0]):

```
st2.w-G1e__hBVDr1CMZJDdfkR-aZMY.FvRzlEVJB_MgCtEKoplLeg.LH-I9_m3p4yw8V_6ZmxsJzaLyqc
```

Decoded id_token data (received from token endpoint; response[3]):

```
{
   "iss": "https://opnap.eu-west-2.elasticbeanstalk.com",
   "exp": 1883377478,
   "iat": 1568017478,
   "aud": "txLT1xNyH_4XccoeWYcTGyc8",
   "auth_time": 1568017163,
   "sub": "b--l-qm5pm5_Bp8-lEvsOb-QuhzrwYMAEMJ5-ooDZoY",
   "nonce": "1568017468",
   "at_hash": "0gqgSyM89h0pUJql17gHmg"
}
```

Decoded userinfo response (returned from userinfo endpoint in exchange for access_token (above)):

```
{
   "sub": "b--l-qm5pm5_Bp8-lEvsOb-QuhzrwYMAEMJ5-ooDZoY",
   "name": "Test Holmes",
   "family_name": "Holmes",
   "given_name": "Test",
   "email": "test@gravitaz.co.uk"
}
```

Dynamic Discovery

Apps that want to use the various OIDC endpoints for authorisation, token exchange, user info and refresh can discover these dynamically through convention - https://opnap.eu-west-2.elasticbeanstalk.com/.well-known/openid-configuration is a JSON file defined by https://openid.net/s pecs/openid-connect-discovery-1_0.html for clients to use.

API Proxy (AWS Lambda)

Server side code is required to support native clients with Gigya since it is bad security practice to deploy apps that contain the client secret.

The API proxy is implemented on AWS using CDK, API Gateway and Lambda. Software is here: https://git.yoox.net/users/ti.holmes/repos/gigyahelpers

To deploy the proxy to AWS:

1. Download/clone the code from https://git.yoox.net/users/ti.holmes/repos/gigyahelpers
2. Configure 'cdk.json' with the appropriate values for:
    a. Gigya API Key
    b. Gigya Client Id/Secret (see above)
    c. API Domain Name
    d. SSL Certificate URN in AWS to match API domain name
    e. Note At present this solution only works with Gigya EU
3. Follow the instructions here to install the AWS CDK software and configure your environment
4. Deploy the proxy using **cdk deploy**

Endpoints

- **POST /sign { consent: <string> } { sig: <signature> }** - signature formed from GIGYA_PARTNER_SECRET using algorithm GIGYA_SIGNATURE_ALGORITHM (default SHA1)
- **any /showConfig { API_KEY: <api_key>, CLIENT_ID: <gigya_client_id> }** provides info on the configured values shown.
- **POST /decode&verify=true|false { token: <token_to_decode> }** - decodes an id_token and optionally verifies the signature against the public key associated with the issuer
- **any /authorize** redirects to the Gigya authorize endpoint, which renders the Proxy Page configured in Gigya as shown above (e.g. opnap.gravitaz.co.uk/RPLogin.php). Gigya expects a redirect_uri with a value that must match a redirect URL defined in the configuration shown above (Edit RP). If a code_challenge and method (PKCE) are supplied, they are stored in a persistent database (dynamodb). The Proxy Page should support an appropriate flow which may include a consent grant.
- **POST /token { token: <token>, grant_type: <authorization_code>, redirect_uri: <uri> }** forwards the request to Gigya, inserting an Authorization header formed from the client ID and secret if not already present. If response_type = code then a code_verifier is expected which must match the code_challenge and method specified on the authorize call (Gigya doesn't currently support PKCE out of the box). The JSON response is returned to the caller, which should include an Access Token plus optionally ID Token and Refresh Token.
- **POST /userinfo**   forwards the request to the Gigya userinfo endpoint. The header should include an appropriate Authorization header including the Access Token provided. Response will be the appropriate userinfo data formed from the access scope negotiated during the authorize flow.
- **POST /refresh { token: <token>, client_id: <client_id>, grant_type: refresh_token, scope: <optional scopes> }** forwards the request to the Gigya refresh endpoint inserting a basic Authorization header if not supplied. Response is a JSON structure with new Access and Refresh tokens.

Note JSON structures returned can be filtered

Mobile Clients

Android

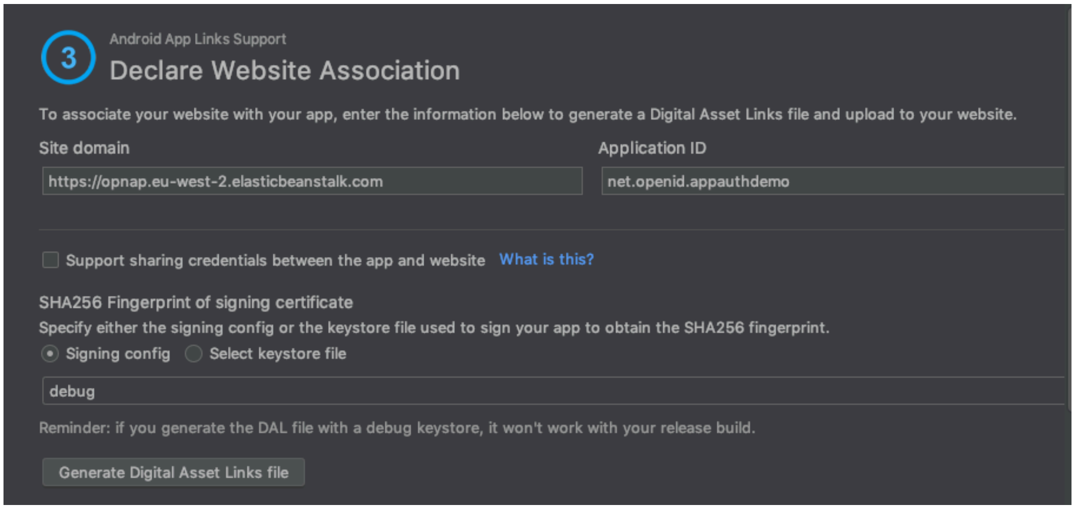
The Android app itself declares the URL that it will accept as an intent from a client opening a URL:

AndroidManifest.xml

```
<activity android:name="net.openid.appauth.RedirectUriReceiverActivity">
    <intent-filter android:autoVerify="true">
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="https"
          android:host="opnap.gravitaz.co.uk"
          android:path="/oidc/oauth-validate.php"/>
    </intent-filter>
</activity>
```

The Android platform requires proof from the domain itself that this action is authorised. It does this by looking for and inspecting a file '/.well-known/assetlinks.json' which must be hosted at the path declared by the app, and protected by a signed and trusted SSL certificate.

The assetslink.json file can be generated from the Android Studio tool used to author/edit the Android app:



Android App Links Support

Declare Website Association

To associate your website with your app, enter the information below to generate a Digital Asset Links file and upload to your website.

Site domain: https://opnap.eu-west-2.elasticbeanstalk.com

Application ID: net.openid.appauthdemo

☐ Support sharing credentials between the app and website What is this?

SHA256 Fingerprint of signing certificate

Specify either the signing config or the keystore file used to sign your app to obtain the SHA256 fingerprint.

● Signing config ○ Select keystore file

debug

Reminder: if you generate the DAL file with a debug keystore, it won't work with your release build.

Android requires the file to be signed.

Sample assetlinks.json file

```
[{
    "relation": ["delegate_permission/common.handle_all_urls"],
    "target": {
      "namespace": "android_app",
      "package_name": "net.openid.appauthdemo",
      "sha256_cert_fingerprints":

["82:9F:89:58:01:BC:66:BF:23:47:6F:06:3A:D6:3A:5F:CE:22:25:7A:73:11:9B:9B:
80:4C:9A:72:E3:D8:FF:ED"]
    }
}]
```

Useful source of info: https://stackoverflow.com/questions/44209477/what-is-the-assetlinks-json-file-for-when-using-android-deep-links

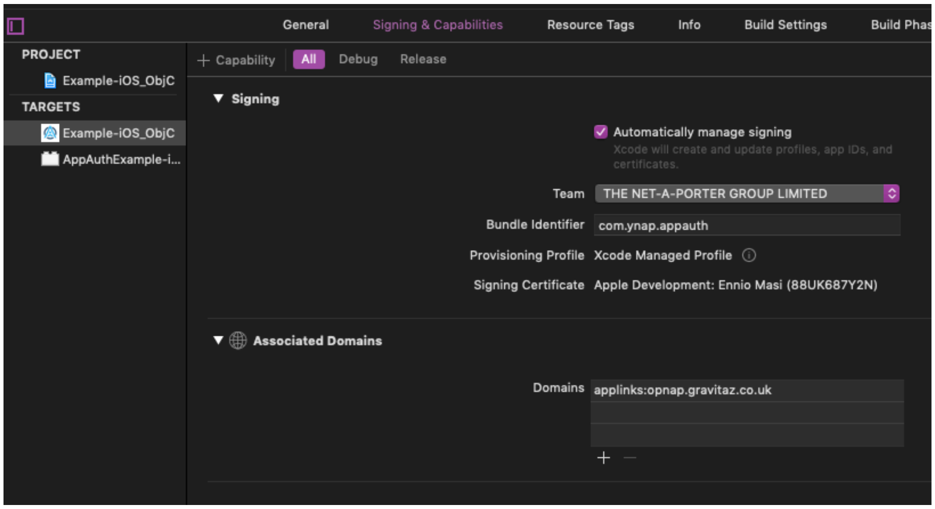
Source code: https://git.yoox.net/users/ti.holmes/repos/appauth-android/browse

iOS

The mechanism for Apple is similar, but also requires the application to be registered through the Apple developer website. For the purpose of this exercise, the app was registered and provisioned through the In Season iOS team, and given a unique App Id. The App was granted appropriate permissions, and signed by the team.

Configuration within Application

Application must enable the Associated Domains capability, define the appropriate domains and intention and provide appropriate certificates for signing the app before it can be distributed. Without this, the redirect back from the Gigya login won't pass control back to the native app, and instead will show a web page containing the authorisation code.



▼ Signing

☑ Automatically manage signing
Xcode will create and update profiles, app IDs, and certificates.

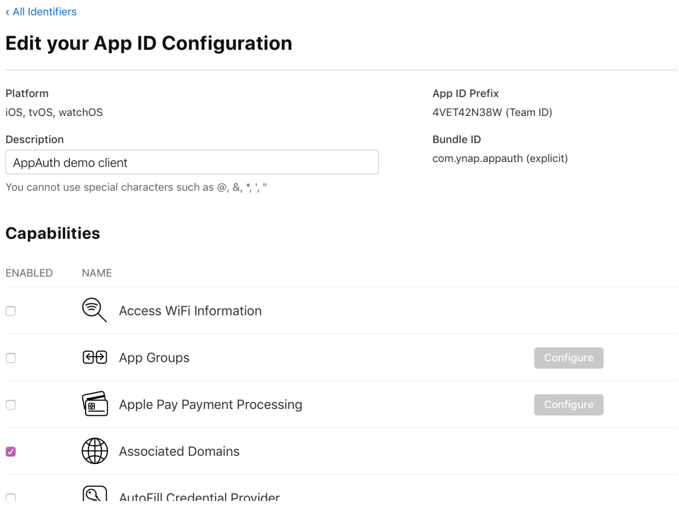
Team: THE NET-A-PORTER GROUP LIMITED
Bundle Identifier: com.ynap.appauth
Provisioning Profile: Xcode Managed Profile
Signing Certificate: Apple Development: Ennio Masi (88UK687Y2N)

▼ Associated Domains

Domains: applinks:opnap.gravitaz.co.uk

Configuration within Apple Developer portal

Application must be granted appropriate permission within the Apple development portal.


‹ All Identifiers

Edit your App ID Configuration

Platform
iOS, tvOS, watchOS

App ID Prefix
4VET42N38W (Team ID)

Description
AppAuth demo client
You cannot use special characters such as @, &, *, ', "

Bundle ID
com.ynap.appauth (explicit)

Capabilities

ENABLED	NAME	
□	Access WiFi Information	
□	App Groups	Configure
□	Apple Pay Payment Processing	Configure
☑	Associated Domains	
□	AutoFill Credential Provider	

Apple App Site Association file

The equivalent to the Android assetlinks.json file is the Apple-App-Site-Association file (or ASAA). This also must be hosted by a site using a trusted SSL certificate (not self-signed), and placed at the /.well-known path.

apple-app-site-association

```
{
  "applinks": {
    "apps": [],
    "details": [
      {
        "appID": "4VET42N38W.com.ynap.appauth",
        "paths": [ "/oidc/oauth-validate.php" ]

      }
    ]
  }
}
```

One slight difference between iOS and Android is that the redirect returned to a web page displayed in a browser will NOT be routed through to the claimed native app unless the hostname is different from that of the triggering page. So if the app opens the URL at host1.domain.com, performs the authentication but then redirects back to host1.domain.com/oid-oauth-validate.php, iOS will NOT open the native app. For this reason we map the redirect to a different subdomain. In our example we have used both opnap.gravitaz.co.uk AND oauth.gravitaz.co.uk.

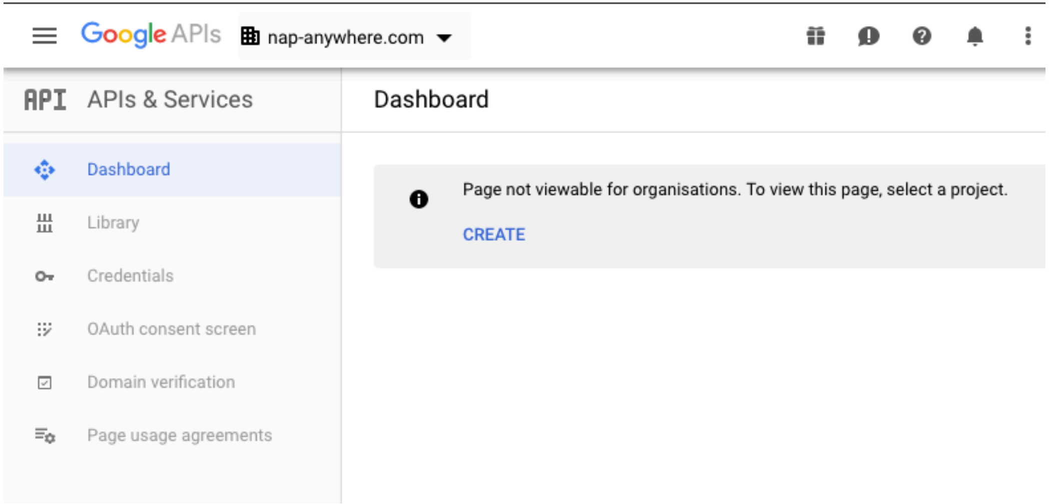
Using Demo App with Google IDP

Following shows how to make the app client work with Google as the IDP

1. Set up Mac for developing native apps - install XCode and Cocoapods
2. Extract code from https://github.com/openid/AppAuth-iOS
3. Follow instructions here: https://github.com/openid/AppAuth-iOS/blob/master/Examples/Example-iOS_ObjC/README.md
4. Set up and run using Google as IDP - https://github.com/openid/AppAuth-iOS/blob/master/Examples/README-Google.md

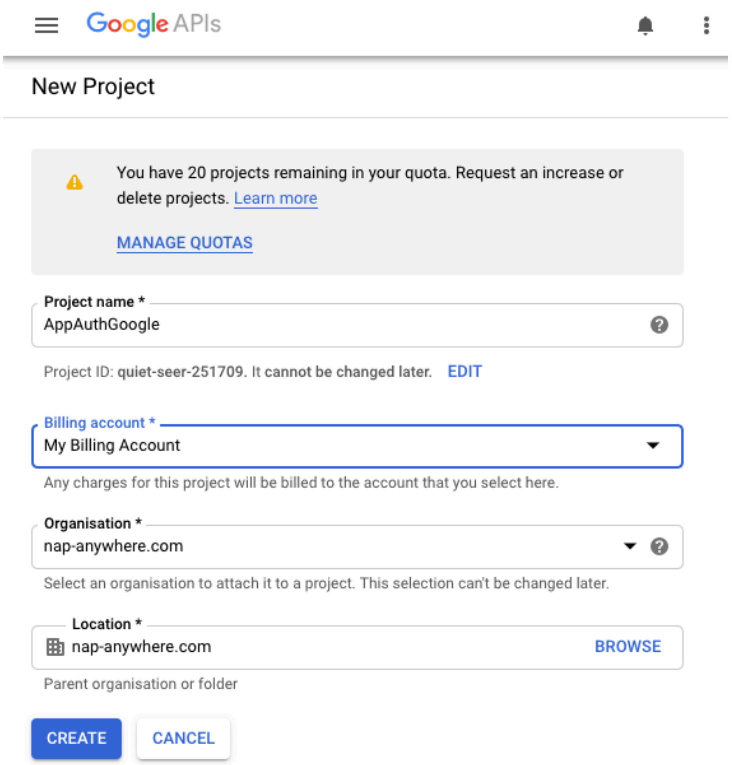
For Google -

1. Log into https://console.developers.google.com using nap-anywhere.com account



Page not viewable for organisations. To view this page, select a project.

2. Create a new project and give it a name e.g. AppAuthGoogle



New Project

You have 20 projects remaining in your quota. Request an increase or delete projects. Learn more

MANAGE QUOTAS

Project name *
AppAuthGoogle

Project ID: quiet-seer-251709. It cannot be changed later. EDIT

Billing account *
My Billing Account

Any charges for this project will be billed to the account that you select here.

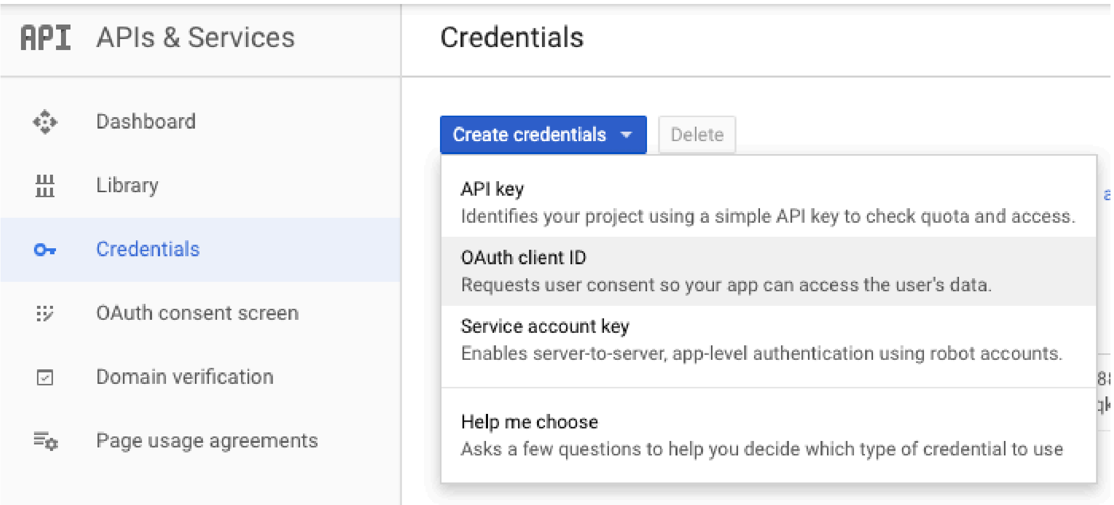
Organisation *
nap-anywhere.com

Select an organisation to attach it to a project. This selection can't be changed later.

Location *
nap-anywhere.com BROWSE

Parent organisation or folder

3. Click on 'Credentials' then 'Create Credentials' and select OAuth Client Id



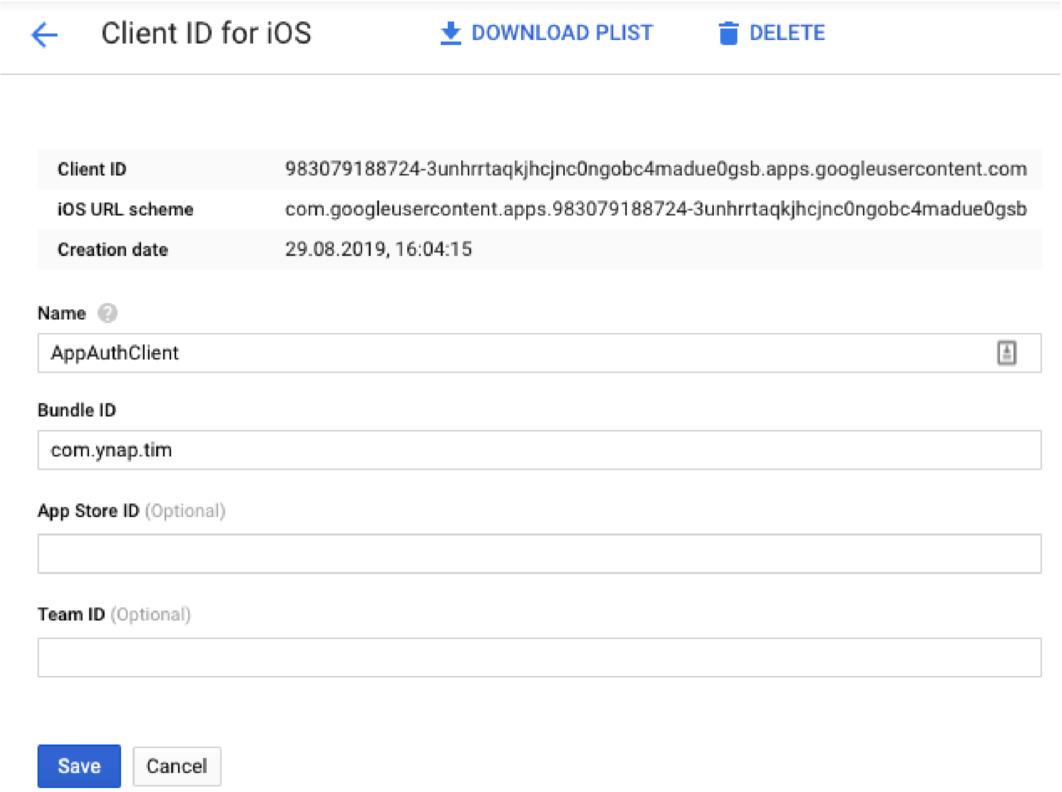
API key
Identifies your project using a simple API key to check quota and access.

OAuth client ID
Requests user consent so your app can access the user's data.

Service account key
Enables server-to-server, app-level authentication using robot accounts.

Help me choose
Asks a few questions to help you decide which type of credential to use

4. Select iOS, then enter a suitable name for the client and also a Bundle ID - ideally a reversed domain name that should be unique.



Client ID	983079188724-3unhrrtaqkjhcjnc0ngobc4madue0gsb.apps.googleusercontent.com
iOS URL scheme	com.googleusercontent.apps.983079188724-3unhrrtaqkjhcjnc0ngobc4madue0gsb
Creation date	29.08.2019, 16:04:15

Name

AppAuthClient

Bundle ID

com.ynap.tim

App Store ID (Optional)

Team ID (Optional)

We now have the following metadata:

Issuer: https://accounts.google.com

Client Id: 983079188724-3unhrrtaqkjhcjnc0ngobc4madue0gsb.apps.googleusercontent.com

Redirect URL: com.googleusercontent.apps.983079188724-3unhrrtaqkjhcjnc0ngobc4madue0gsb:/oauth2redirect/google

Note for this exercise we use 'Custom URI's as the redirect mechanism. These are not supported by Gigya (deemed insecure), which is why we used Universal Links.

Customer Data Access

Using Open ID Connect, a customer can select what attributes of their profile is to be made available to the consuming client during the Consent phase of the OAuth flow. Gigya provides two schemas for customer data - Profile and Data. Profile is a fixed schema, and Data is a flexible extensible schema. Gigya can be configured to support consensual access to profile data claims through the use of custom claims and scopes. A custom claim consists of a field name and a mapping to a profile or data attribute. A custom scope groups custom claims into a single consent.

For example the current MRP Gigya data schema consists of the following:

1. Brand
2. EnterpriseCode
3. RegistrationCode
4. PersonTitle
5. PreferredLanguage

Limitations of Gigya's OIDC

Gigya's implementation of OIDC is limited in the following ways:

1. Access Token cannot be used with other Gigya APIs - only userinfo endpoint
2. No direct integration with Gigya EPM (preference management solution)
3. No native implementation of PKCE - the proxy layer implemented provides limited support but not scalable

We mitigate some of these shortcomings by routing Gigya API calls through AWS. This allows us to:

1. If we can validate the OAuth access token we can potentially route Gigya APIs via a Lambda Authoriser - work in progress
2. Extend the Gigya Screen sets to cover 2 - this worked fairly seamlessly and can be considered done
3. Implement PKCE in the API layer (not ideal as we don't see the authorisation code, so we can't do an indexed lookup in the PKCE_VERIFIER dynamo db table) - this is also done.

Beyond OIDC

The Identity Token, until it expires, is proof of authentication by the customer with the CIAM subsystem.

The API Gateway includes a new API endpoint, /assertAWS. If a valid ID_TOKEN is POSTed to this endpoint, it will:

1. Associate the Gigya identity with an AWS Cognito Federated Identity
2. Return temporary credentials with access rights (IAM Role) associated with the specified Cognito Identity Pool
3. These credentials can be used to sign AWS API calls

TODO: provide an end to end demo scenario