Postcode Application Operation Manual

(Version 1.0.1)

Table of Contents

[Version history 3](#_Toc67239476)

[Pre-requisites 4](#_Toc67239477)

[Authorization 4](#_Toc67239478)

[Setup 4](#_Toc67239479)

[H2 database 15](#_Toc67239480)

[Swagger UI page 17](#_Toc67239481)

[Endpoints 18](#_Toc67239482)

[Get a list of suburbs matching the postcode 18](#_Toc67239483)

[Get a list of postcodes matching the suburb 19](#_Toc67239484)

[Add a new suburb and postcode combination 20](#_Toc67239485)

[Appendix 22](#_Toc67239486)

[Component URLs 22](#_Toc67239487)

# Version history

|  |  |  |  |
| --- | --- | --- | --- |
| Editor | Version | Description | Remark |
| Patrick Cheung | 1.0.0 | Initial release |  |
| Patrick Cheung | 1.0.1 | Amend deployed EC2 instance domain name to a placeholder |  |

# Pre-requisites

In order to use the APIs, you should have a client application which will call the API endpoints.

You can test the endpoints with an API testing tool like postman. The Swagger UI bundled with the Postcode Application also readily provides the ability to test on the endpoints.

## Authorization

For secured endpoints that requires authorization (e.g. when adding a new postcode-suburb combination), you will need to provide authorization details when calling the endpoint.

Your client application should be designed to follow the OAuth2 "authorization code flow" such that you will first obtain an authorization code from the authentication server, and then you exchange this authorization code with an access token. The access token should then be included in the authorization header when calling the secured endpoint.

### Setup

For demonstration purpose, Okta is used as the identity provider. An Okta application has been created to act as the authentication server.

**Graphical user interface, application

Description automatically generated**

We will make use of a debugger tool at https://oidcdebugger.com

It simulates the authentication code flow.

Fill in the details on the tool as follows

|  |  |
| --- | --- |
| Authorize URI | https://dev-27548664.okta.com/oauth2/default/v1/authorize |
| Redirect URI | https://oidcdebugger.com/debug |
| Client ID | 0oacdfazegsY2RsWr5d6 |
| Scope | openid |
| State | Anything, e.g. "abc" |
| Nonce | Anything, e.g. "abc" |

Click on "SEND REQUEST". It will prompt you to the identity provider login page (Okta in this case). Enter the details as follows

(Note: In reality the client application should have registration flow setup with the identity provider. But for now we will just use a pre-created user)

|  |  |
| --- | --- |
| Username | patrickpycheung+test@gmail.com |
| Password | @yMXhLq@Z)28ay<V |

The initial authentication should be successful, and you should see something similar to below

Graphical user interface, text

Description automatically generated

Copy the authorization code.

We will then exchange the authorization code with an access token.  
Using Postman or other tools, make the following request

Required parameters:

|  |  |
| --- | --- |
| grant\_type | authorization\_code |
| code | The authorization code obtained previously |
| client\_id | 0oacdfazegsY2RsWr5d6 |
| client\_secret | FV46QRv5HfR9VTNgaWw-1Sx8qfS56WHDoG8dOzu\_ |
| redirect\_uri | https%3A%2F%2Foidcdebugger.com%2Fdebug |

Same cURL command:

* Sample request

curl --location --request POST 'https://dev-27548664.okta.com/oauth2/default/v1/token?grant\_type=authorization\_code&code=p7O4PlbIVLwUHUsDhAkXN5Cy3qlfw\_CXtnb0vPGte1M&client\_id=0oacdfazegsY2RsWr5d6&client\_secret=FV46QRv5HfR9VTNgaWw-1Sx8qfS56WHDoG8dOzu\_&redirect\_uri=https%3A%2F%2Foidcdebugger.com%2Fdebug' \

--header 'Content-Type: application/x-www-form-urlencoded' \

* Sample response

{

"token\_type": "Bearer",

"expires\_in": 3600,

"**access\_token**": "**eyJraWQiOiJNMVZZQ2tKX183cjBCcGY3a1pSaXZEWFI5M1VHMUlQcXhuTzhhUm4zSDk4IiwiYWxnIjoiUlMyNTYifQ.eyJ2ZXIiOjEsImp0aSI6IkFULmFhN3BfQmsycGRQTElIRFQxR2Jfb1BuQU5GVVotbHRKZmRXVkdrOC14dkUiLCJpc3MiOiJodHRwczovL2Rldi0yNzU0ODY2NC5va3RhLmNvbS9vYXV0aDIvZGVmYXVsdCIsImF1ZCI6ImFwaTovL2RlZmF1bHQiLCJpYXQiOjE2MTYyOTg0NDgsImV4cCI6MTYxNjMwMjA0OCwiY2lkIjoiMG9hY2RmYXplZ3NZMlJzV3I1ZDYiLCJ1aWQiOiIwMHVjZGZyaGM1SnlBdU5zUTVkNiIsInNjcCI6WyJvcGVuaWQiXSwic3ViIjoicGF0cmlja3B5Y2hldW5nK3Rlc3RAZ21haWwuY29tIn0.HJuupnjiMu4OO0sIqibxauvSUsZAZUB1\_kEP0EK8MKtoyje9iQsso5MhVoJKQ3iKr2lJ4Zqyr1viZGhEaC\_tId22z-Dg5bLcD8x-bRWuRyvZRJvjcRP-DD2QxpXBL4dVMyhZRdBcFDuEp15-BFIYyNo32UegY0GGM4KzA16uprGKj2aaawJeynLk3GYnKmKuCmiBf2DWC3LCpR1pK1OmPX1DuZnZc0tV4wxHI7yREVdFrOl0xhtkId-Qoe3VKR2OEPgi3CFd4eQqebIn-wxNakkFaPkgCi4kUlFo8IvWsVUlgOlOqMu2f5IbdP2EAzQToganUhZetYi-PTH0ChpycA**",

"scope": "openid",

"id\_token": "eyJraWQiOiJNMVZZQ2tKX183cjBCcGY3a1pSaXZEWFI5M1VHMUlQcXhuTzhhUm4zSDk4IiwiYWxnIjoiUlMyNTYifQ.eyJzdWIiOiIwMHVjZGZyaGM1SnlBdU5zUTVkNiIsInZlciI6MSwiaXNzIjoiaHR0cHM6Ly9kZXYtMjc1NDg2NjQub2t0YS5jb20vb2F1dGgyL2RlZmF1bHQiLCJhdWQiOiIwb2FjZGZhemVnc1kyUnNXcjVkNiIsImlhdCI6MTYxNjI5ODQ0OCwiZXhwIjoxNjE2MzAyMDQ4LCJqdGkiOiJJRC5ZSE5ROXJPb3JxYl92UDMyQkl6R212Y242T1g1WWZ0RUVQRU5JY2ZjekdJIiwiYW1yIjpbInB3ZCJdLCJpZHAiOiIwMG9hM3pmMm83UnlIZ1cybzVkNiIsIm5vbmNlIjoiZjMwZWFqaG53amIiLCJhdXRoX3RpbWUiOjE2MTYyOTgwNTQsImF0X2hhc2giOiJyX0FLWElDOWFpMEdmYi15TjJDbkRBIn0.GsRqqK-XhEzayEKDu84vu9sMxHqL\_SWTURUNYNVcG2rcmurKB4\_RCQiSk2rrbY0k6UgWCxwII\_Wo6-vFCUdcYaUFOCabk-wgkdT1CyCx6XOqBexZPEOM6AcC9-7y462CHCaVgsmVvbjDVMgbbCXB--oT9o6ix1e-Nk\_rGV5NvZklokFvjll5oB2UG4zKMKu07RaQf244LVb0p0UUUrIQ3elmaauaYVkojsbGVgl-Wm4VPcqpjiYE2Yd35McGhSiiP4j72BkPAftV\_O3prDc8g4LAPvwGq0qTOU3hI16lZqxjEuwyY25-y8vNztSUAffotdystTMTY9Sny2aoGw1a2Q"

}

Record the access token, which is a JWT token.

Later on, when calling secured endpoint, we will need to include the JWT token in the "Authorization" header.

Postman example:

Graphical user interface, text, application

Description automatically generated  
  
Graphical user interface, text, application

Description automatically generated

Swagger UI example:

Configure the use of authorization

Graphical user interface, application, Teams

Description automatically generated

Paste the JWT token (Note: add the word "Bearer " before it)

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Then all requests afterwards will include the JWT in the "Authorization" header.

Graphical user interface, text, application

Description automatically generated

Note:

Remember to logout from the authorization if you are testing on general endpoints that do not require authentication

Graphical user interface, application

Description automatically generated

# H2 database

You can access the H2 admin console at the following URL, where you can prepare or modify the data for testing.

Local deployment:

http://localhost:8080/h2-console

Cloud server deployment

http://<Cloud server URL>:8080/h2-console

Enter the credentials as below to login the console

|  |  |
| --- | --- |
| JDBC URL: | jdbc:h2:mem:testdb |
| User Name: | sa |
| Password: | <BLANK> |

e.g.

http://ec2-100-25-4-203.compute-1.amazonaws.com:8080/h2-console

Graphical user interface, application, Word

Description automatically generated

Text

Description automatically generated

# Swagger UI page

You can access the H2 admin console at the following URL, where you will find descriptions about the API endpoints. It also provides the ability to test the endpoints as an API client.

Local deployment:

http://localhost:8080/swagger-ui.html

Cloud server deployment

http://<Cloud server URL>:8080/swagger-ui.html

e.g.

http://ec2-100-25-4-203.compute-1.amazonaws.com:8080/swagger-ui.html

Graphical user interface, text, application, email

Description automatically generated

# Endpoints

## Get a list of suburbs matching the postcode

|  |  |
| --- | --- |
| Path | /api/postcode/suburb |
| Verb | GET |
| Description | This endpoint allows the user to get a list of suburbs matching the postcode pattern. |
| Query parameter | The postcode pattern.  It must be of at least 3 but not more than 4 characters.  It can only be a number.  e.g.  "3121", "312" |

Sample request:

Graphical user interface, application

Description automatically generated

## Get a list of postcodes matching the suburb

|  |  |
| --- | --- |
| Path | /api/postcode/postcode |
| Verb | GET |
| Description | This endpoint allows the user to get a list of postcodes matching the suburb pattern. |
| Query parameter | The suburb pattern.  It must be of at least 3 but not more than 100 characters.  The query is case-insensitive.  e.g.  "PARRAMATTA, NSW", "Par" |

Sample request:

Graphical user interface, application

Description automatically generated

## Add a new suburb and postcode combination

|  |  |
| --- | --- |
| Path | /api/postcode |
| Verb | PUT |
| Description | This endpoint allows the user to add a new suburb and postcode combination |
| Request body | The postcode and suburb information in JSON format.  e.g.  {  "postcode": "1234",  "suburb": "Test"  } |
| Remark | Only if the combination is considered new will the entry be actually added.  2 entries are the same if they have the same postcode and suburb (case-insensitive). For example, if the user attempts to add a combination { "postcode": "1234", "suburb": "Test"}, and the DB already has a record with postcode="1234" and suburb="test", the insert will not be actually carried out , although it will still appear to be successful to the user. |

Sample request:

Graphical user interface, application

Description automatically generated

# Appendix

## Component URLs

The application has been deployed to an AWS EC2 instance for demonstration purpose. The links to the function components are as below

|  |  |
| --- | --- |
| Health-Check page | http://[*EC2 instance domain*]:8080/healthcheck.html |
| Endpoint to get a list of suburbs matching the postcode. | http://[*EC2 instance domain*]:8080/api/postcode/suburb |
| Endpoint to get a list of postcodes matching the suburb. | http://[*EC2 instance domain*]:8080/api/postcode/postcode |
| Endpoint to add a new suburb and postcode combination. | http://[*EC2 instance domain*]:8080/api/postcode |
| H2 database admin console | http://[*EC2 instance domain*]:8080/h2-console |
| Swagger UI page | http://[*EC2 instance domain*]:8080/swagger-ui.html |

Note:

The EC2 instance domain will change upon server restart. In order to keep a static IP, an elastic IP address needs to be allocated (not in free tier)