**Accessing okta APIs using postman collection**

Press enter or click to view image in full size

A blue background with black text

AI-generated content may be incorrect.

Okta is an identity and access management(IAM) platform for Workforce Identity as well as Customer Identity(CIAM). Which provides products such as SSO, Universal Directory, Advanced Server Access, Access Gateway, MFA, Lifecycle Operations etc., as well as services such as Directories, Integrations, Identity Engines, Workflows etc..

Okta provides user interface as well as APIs to configure the environment. Graphical user interface is mainly for the okta administrators to configure and monitor the okta environment such as creating users and applications, performing security configurations, integrating SSO, API security, access management etc. It provides APIs to perform the same operations programatically, as well as to authenticate and authorise from and to other applications.

In this article I will introduce some of the important okta APIs and explain how to access those APIs from postman.

**The key okta APIs**

It is well [documented](https://developer.okta.com/docs/reference/core-okta-api/) here about user sign in APIs as well as okta object management APIs. Okta provides the [postman collection](https://developer.okta.com/docs/reference/postman-collections/) of most of the APIs. The user APIs are public or protected by OAuth2, OIDC etc. Most of the okta management APIs are protected by okta API Token.

**Creating API Token for okta API access**

Login to okta console using the relevant administrator, and go to API -> Tokens -> Create Token. You can give any name for this token, but I would choose to give the name based on the use case.

Press enter or click to view image in full size

A screenshot of a computer

AI-generated content may be incorrect.

Create API Tokens window — okta console

Once you have created the token, copy and store the Token Value securely before you close the popup window, because this is the only time the Token Value is shown for us to copy.

A screenshot of a computer error message

AI-generated content may be incorrect.

okta API token created window

The token is valid and active until the **admin user who created is active** and idle time of the token usage is less than **30 days**. Make sure that the token is used frequently, and it is best practice to create a dedicated admin user for each service with only required permissions and create API Tokens by that admin user, rather than with a common or super admin user.

You can notice that the Expiry time for the token is set to 30 days after the created/Last Used date. We can check the Expiry time gets updated based on the Last Used date after we start consuming the API Token.

A screenshot of a computer

AI-generated content may be incorrect.

List of API Tokens in okta console with the details

**Download postman collection for Users API**

Download the [okta users postmam collection](https://developer.okta.com/docs/reference/postman-collections/) and open it in your postman, either in web or app, you should be able to see the API collection in your postman as shown in the below screenshot.

A screenshot of a computer

AI-generated content may be incorrect.

Postman collection for okta’s Users APIs, and GET user API is opened in the execution window

**Execute Users API from postman**

To begin with, I am going to execute the *GET {{url}}/api/v1/users/{{userId}}* API, which is the API to retrieve a user’s details from okta using the userId. For this we need to configure the required parameters and the API token. To configure the required parameters and API Token, we can either replace the placeholders or configure postman **environment variables,** so that we can use the same attributes/configuration for other APIs or switch between environments without having to update the APIs every time.

We need to configure the following attributes in order to consume the above API.

1. **url** — URL of the okta domain from where we want to retrieve the user details.  
2. **userId** — user’s loginId or userId which we want to retrieve.  
3. **apikey** — the API Token created in the same okta environment from where we want to retrieve the user details, make sure that API Token is created for the same purpose to make sure that the API Token has the required permission to perform the operation.

Let’s configure the attributes in the postman environment. I created a new environment called demo, but based on your use case you can create one for each environment in your organisation such as dev, staging and prod.

A screenshot of a computer

AI-generated content may be incorrect.

postman environment variables are configured to use for okta users API against the demo environment

You might have noticed that I used the email address for the userId, because in this case, that is the loginId of the user I want to retrieve. We can alternatively use the okta userId starting with **00u.**Please note that only users collection allows us to use the **loginId** in place of **userId**, and in all other APIs it is the must to use okta userId starting with **00u**.

Make sure to save your changes and select the right environment before executing the API for the values from the environment variables to reflect in your APIs when you execute them.

Now we are ready to execute the API to retrieve the user details.

A screenshot of a computer

AI-generated content may be incorrect.

Sucessful retrieval of user details using the okta API configured in postman

If you have noticed the parameters configured in the API is replaced with the values we configured in the environment variables in postman.

A screenshot of a computer

AI-generated content may be incorrect.

Postman console view to demonstrate the request parameters with the values configured in the postman environment variables

Before we wind up, let’s look at the API Token in the okta console whether the last Expiry Time is changed after we used the API token.

A screenshot of a phone

AI-generated content may be incorrect.

Okta API Token with the updated Expiry time

You can notice here that the Expiry time is updated based on the Last Used time, and which is 30 days since Last Used time.