

Kronos API Example in Dimensions – Workload Templates

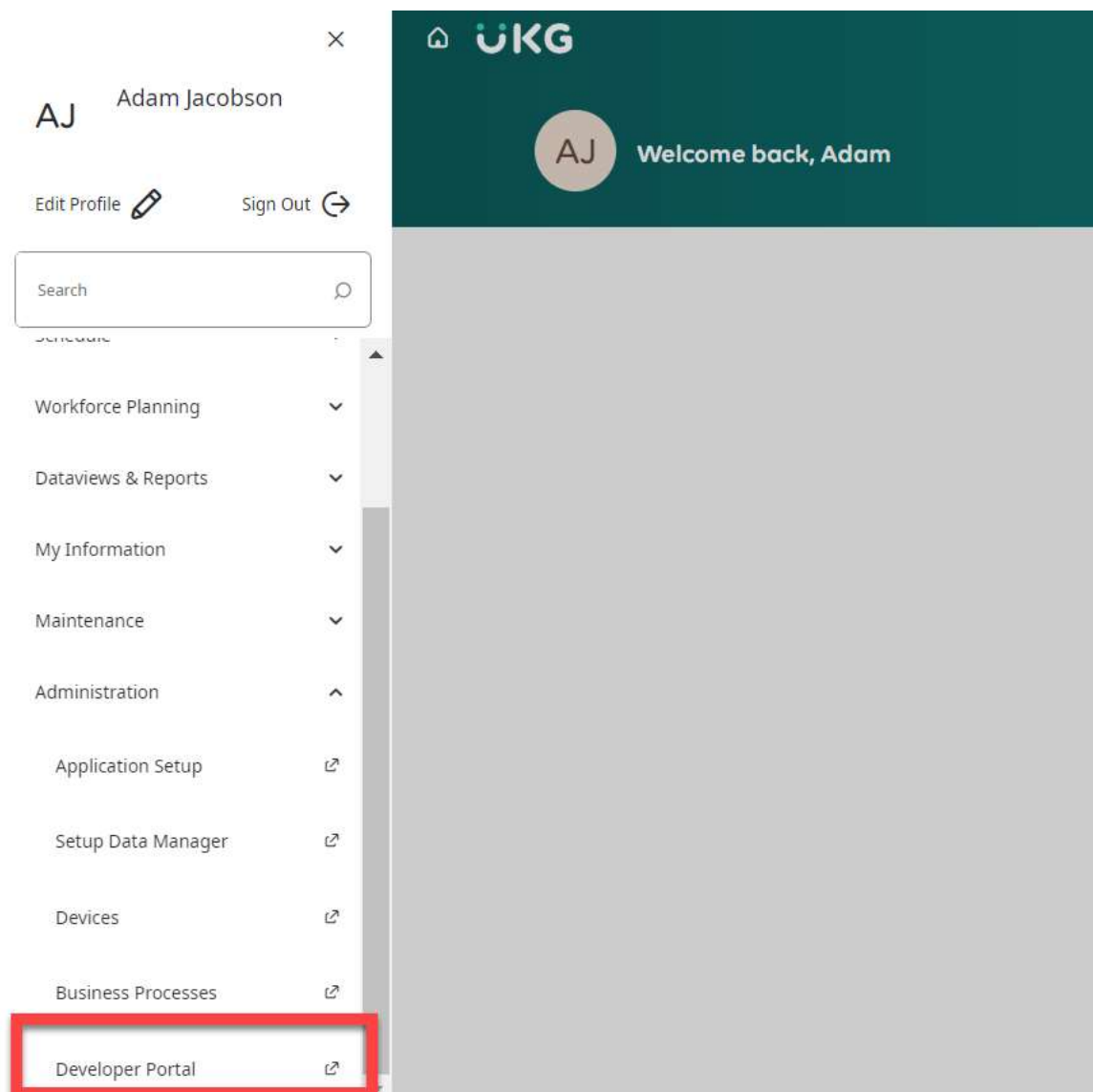
This post will take you through a step-by-step Dimensions Rest API example in Postman. In our Postman example, we'll be sending two requests:

- 1) Access Token – This retrieves the access token which we need to run the next request.
- 2) Retrieve Person by Extension – This retrieves data about an employee.

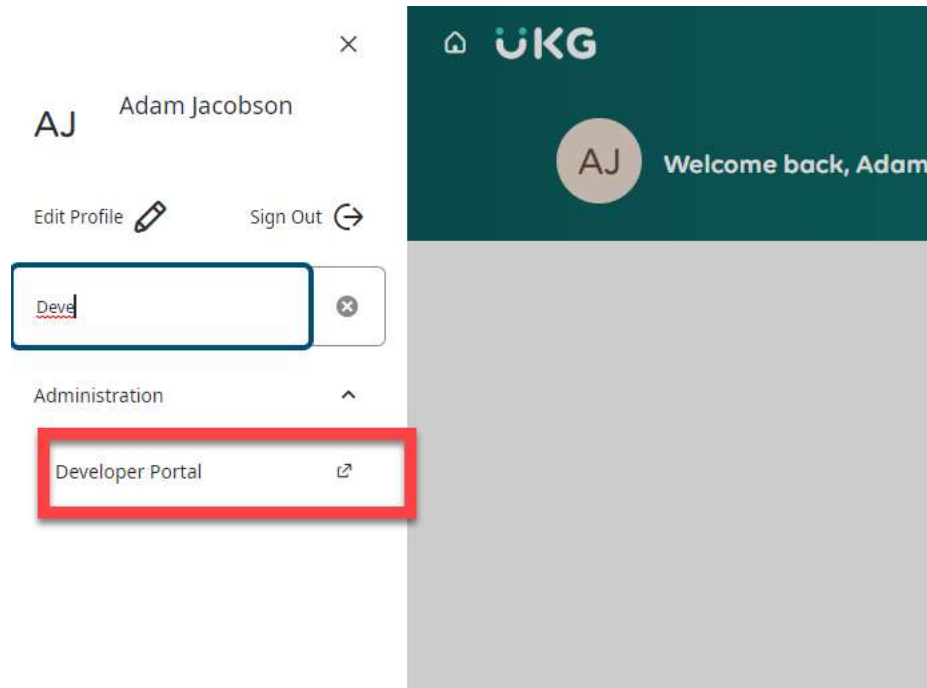
If you just want the example, feel free to go directly to the code in GitHub.

We're not going to repeat what's available in the documentation. If you're new, you may wonder to get to the documentation. The answer is the developer portal.

You can find the Developer Portal under administration:



Or by searching:



If the developer portal does not appear, have your administrator confirm that your Function Access Profile has the appropriate access. The required access can be found under: Manager Common Setup -> Developer Portal -> Administration Access and Developer Access.

Manager - Department Manager						Various	?
Manager - Common Setup						Various	?
+ Hyperfind						Various	?
+ People Editor						Various	?
Organizational Set Editor	Allowed	Allowed	Allowed	Allowed		Various	?
+ Jobs and Business Structure						Allowed	?
Worker Type	Allowed	Allowed	Allowed	Allowed		Allowed	?
+ Event Manager						Allowed	?
Table Import						Disabled	?
Transaction Assistant		Allowed	Allowed	Allowed		Allowed	?
+ Summary						Various	?
Schedule Configuration						Allowed	?
Hyperfind Profiles						Allowed	?
+ Process Management Setup						Allowed	?
Calendar Views Setup						Allowed	?
Delegate Profiles						Allowed	?
Override Alert Event Configuration						Allowed	?
Role Profiles						Various	?
+ Integrations						Various	?
Workforce Absence Setup						Disabled	?
Dataloader Management	Allowed	Allowed	Allowed			Allowed	?
+ Schedule Group Edit						Allowed	?
Design Integrations						Allowed	?
Developer Portal						Allowed	?
Administration Access						Allowed	?
Developer Access						Allowed	?

Please note that the account you'll be using will need appropriate access to do whatever you'd like to do. You can't look up an employee (as we'll do in this example) if you don't have access to the employee from within the application.

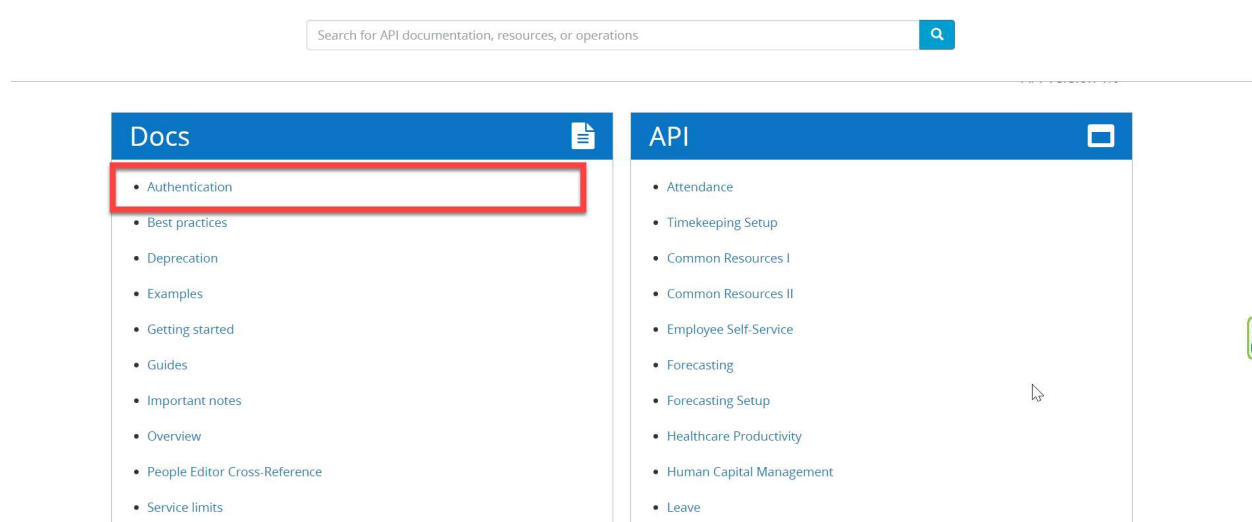
Now let's show you how to find the documentation for our requests:

Dimensions Developer Portal Documentation

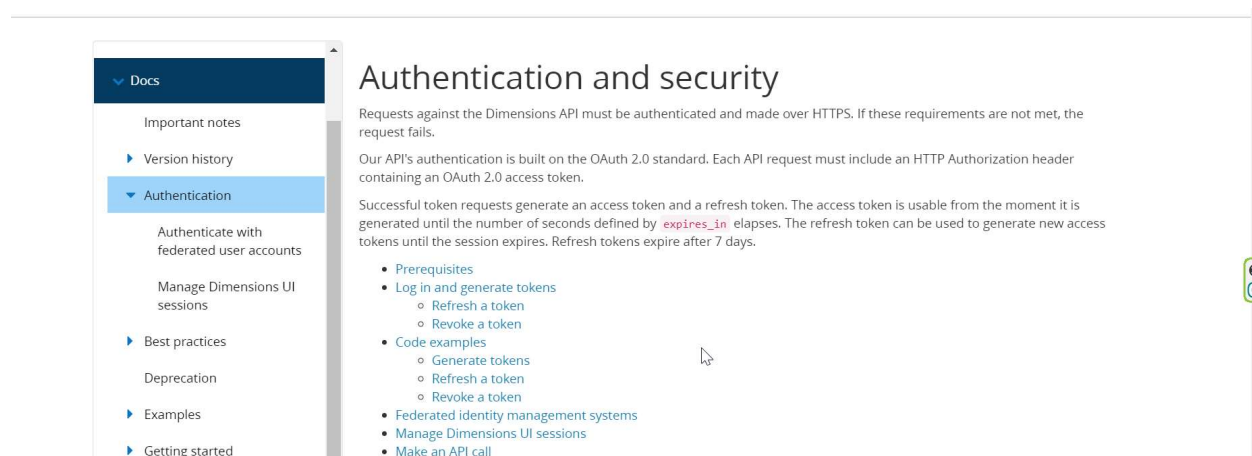
Step 1 – Authentication:

Unlike UKG Pro/UltiPro APIs which we've described [here](#), Dimensions APIs require a two-step process. The first request gets you an access token which is then used in the second request..

To find the documentation from the developer portal, click on Authentication.

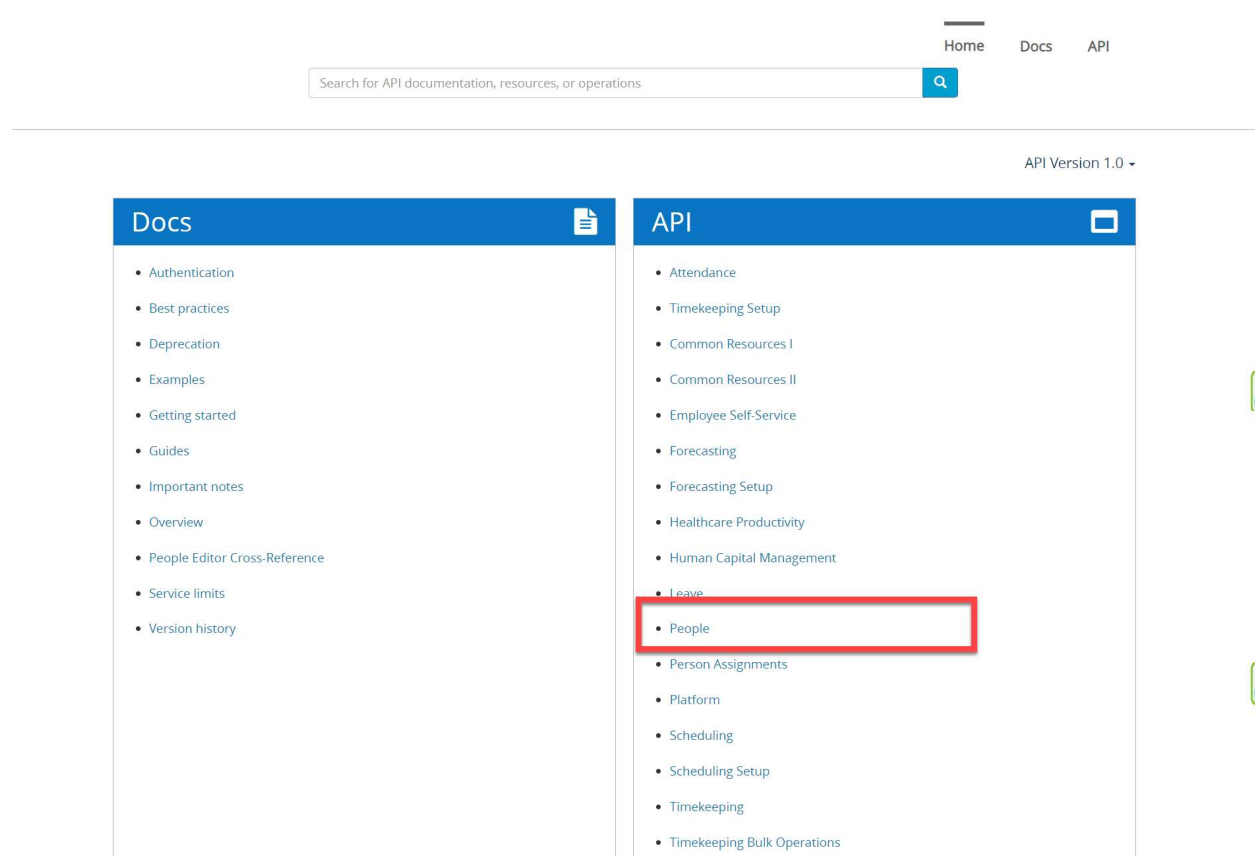


Again, I do suggest you read the documentation. We're focusing on our particular Postman example. ([And aren't you missing SQL and your on-premise database already!](#))

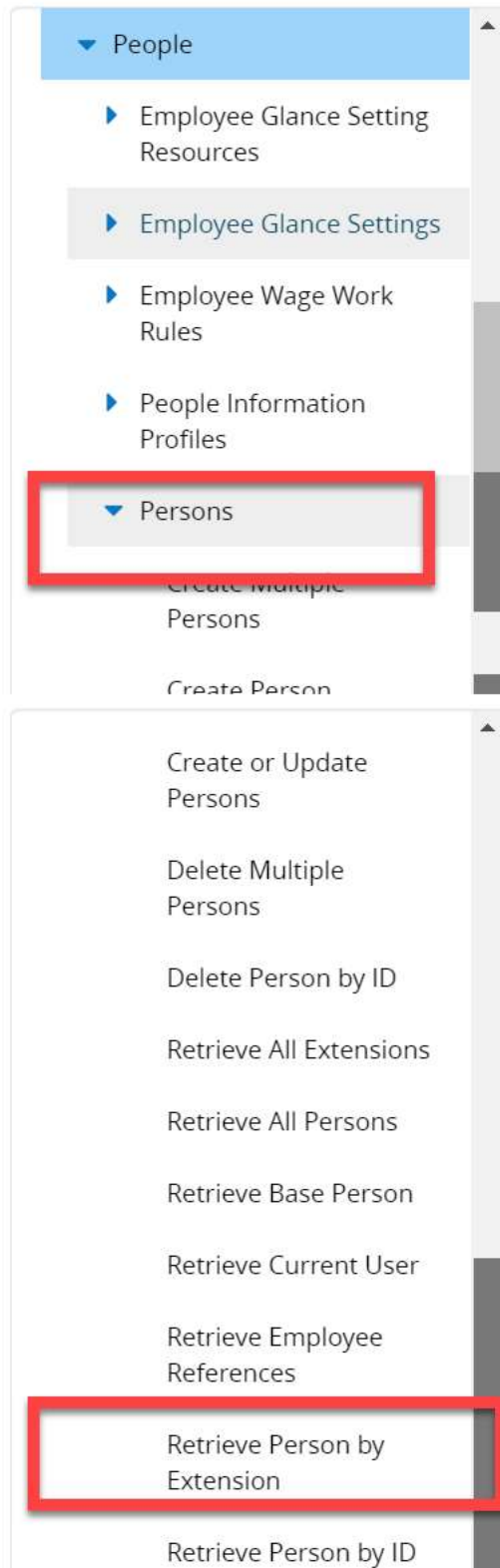


Step 2 – Retrieve Person by Extension

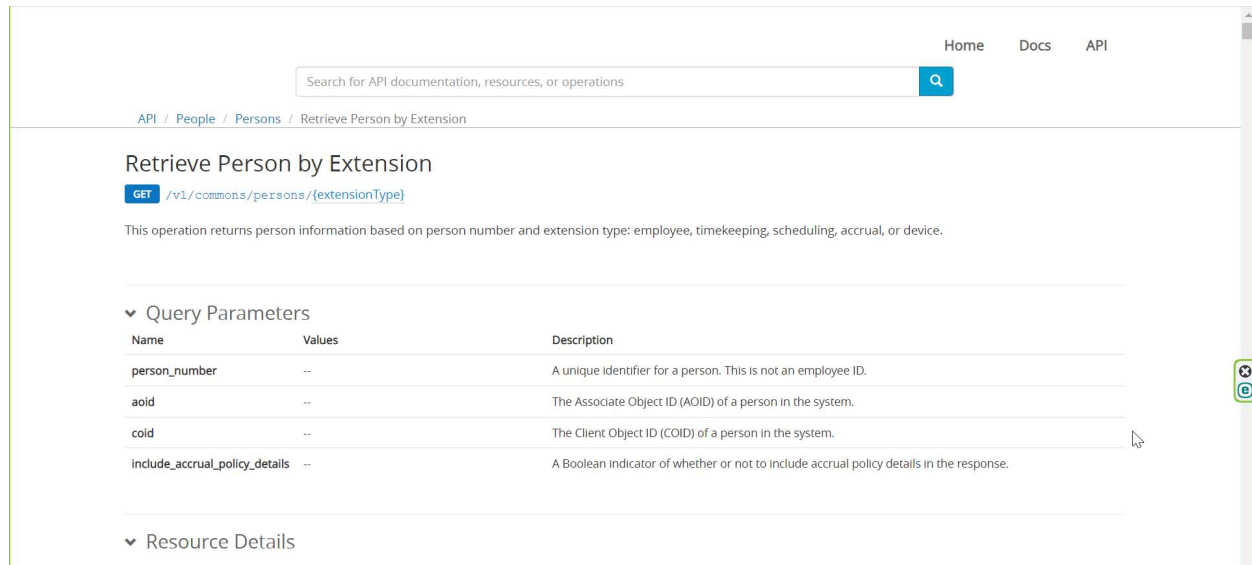
We're going to use a simple API which retrieves data for a particular person. To find the documentation, click under People under the API column.



Now, expand “Persons” and scroll down until you see “Retrieve Person by Extension”



And you'll find the documentation:



Please note:

You'll find a search function at the top. However, I've found that it returns so many results that it's at best "sub-optimal".

Postman Example:

Dimensions Authentication/access_token

Now to postman.

For the first part

We'll be looking at four parts of the postman request.

The URL

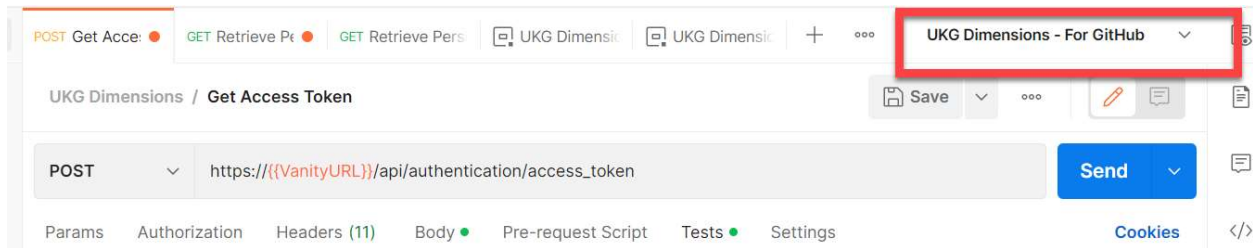
Headers

Body

Test

For this example, everything that will be custom to your site is stored in an environment variable.

The environment will be called UKG Dimensions – For GitHub:



Here are all the variables we'll be using:

A screenshot of the 'UKG Dimensions - For GitHub' collection in Postman. It shows a table of variables with columns for 'Variable', 'Type', 'Initial value', and 'Current value'. All variables are checked and have a default type. The 'person_number' variable has an initial value of '123456'.

	Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/>	VanityURL	default	VanityURL	
<input checked="" type="checkbox"/>	client_id	default	client_id	client_id
<input checked="" type="checkbox"/>	client_secret	default	client_secret	client_secret
<input checked="" type="checkbox"/>	UserName	default	UserName	UserName
<input checked="" type="checkbox"/>	Password	default	Password	Password
<input checked="" type="checkbox"/>	appkey	default	appkey	appkey
<input checked="" type="checkbox"/>	person_number	default	123456	123456
<input checked="" type="checkbox"/>	AccessToken	default		
	Add new variable			

We'll review each of these in depth. However, in brief:

The first three values were provided during implementation. If you don't have them, you can open a case to get them:

- VanityURL
- Client_id
- Client_secret

The next two are for the user that you will use to run the API

- UserName
- Password

The appkey will need to be setup within the application. We'll walk this through further in the post.

The Person_number can be any valid employee number in your application TO WHICH the username has access.

The AccessToken is a variable to pass the value from the authentication request to the second request.

The URL

The URL tell us the API you wish to call and the type of API (POST, GET, etc).

Please make sure you pay attention to the request type. Some request which are just reading data use POST. (As I figured out in the middle of a support call. Oops.)

The screenshot shows the 'UKG Dimensions / Get Access Token' interface. The request method is 'POST' and the URL is 'https://{{VanityURL}}/api/authentication/access_token'. The 'Headers' tab is selected, showing a table with two headers: 'Content-Type' with value 'application/x-www-form-urlencoded' and 'appkey' with value '{{appkey}}'. Both headers are checked. The 'appkey' value is highlighted with a red box.

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/x-www-form-urlencoded	
<input checked="" type="checkbox"/> appkey	{{appkey}}	

The request needs the vanity URL. The vanity URL should be provided by the implementation team. Otherwise, you'll need to open a case. (It took very little time to receive the information)

Headers

You only need two values in the headers. The content type is standard.

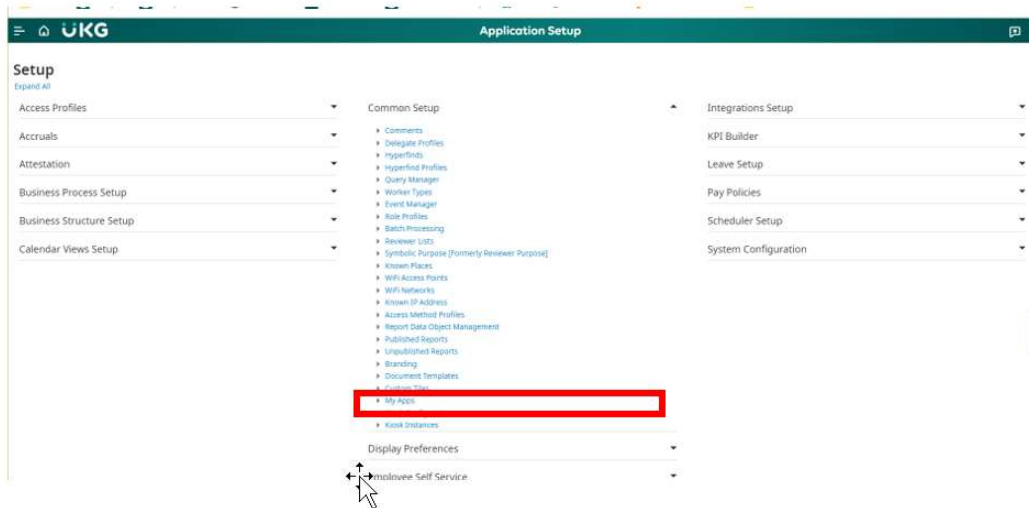
This screenshot is identical to the one above, showing the 'UKG Dimensions / Get Access Token' interface with the 'Headers' tab selected. The 'appkey' value in the header table is highlighted with a red box.

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/x-www-form-urlencoded	
<input checked="" type="checkbox"/> appkey	{{appkey}}	

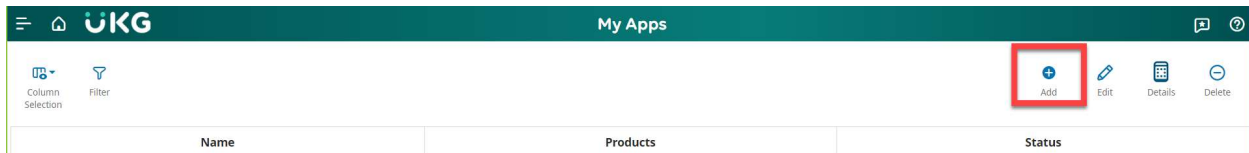
You need to set up the AppKey in the Dimensions application.

To do so, go to Application Setup -> Common Setup-> My Apps

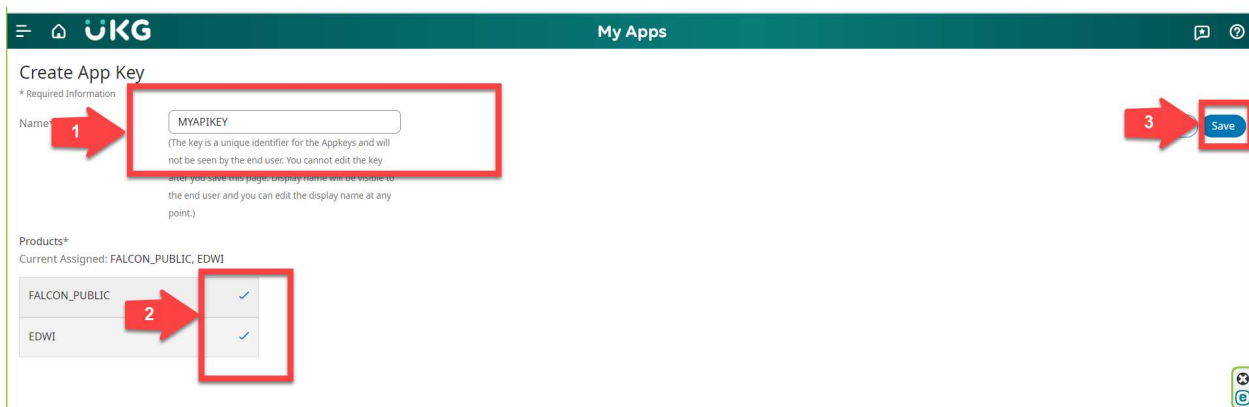
Page 27



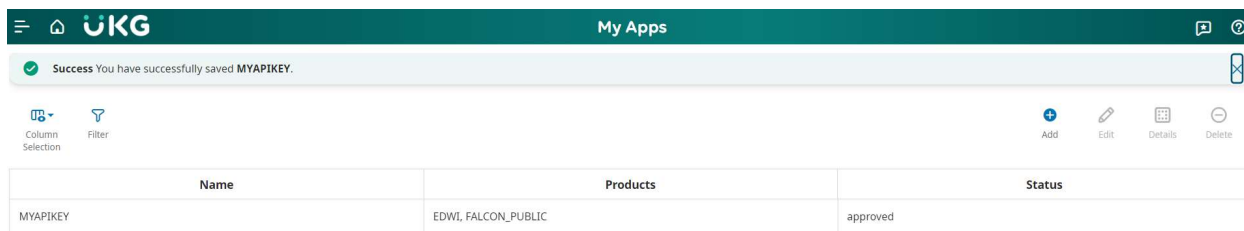
Then click Add:



- 1) Name your key. The name doesn't matter and won't be used anywhere else. (It can be called Fred. Not that I suggest that you do so.)
- 2) Check what products this key may use.
- 3) Click Save.



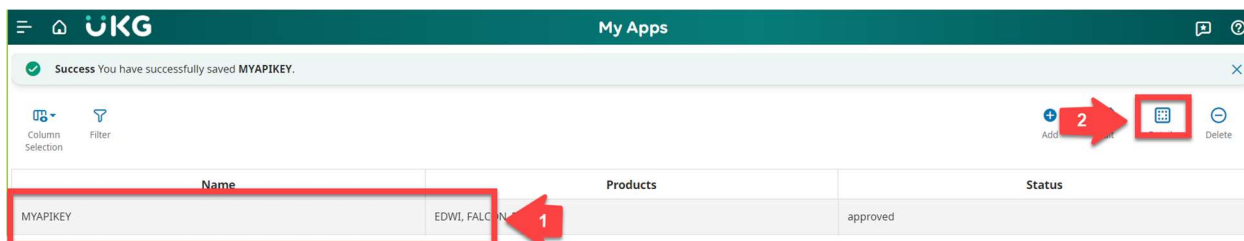
You should receive a message that the key was successfully created.



The screenshot shows the UKG My Apps interface. At the top, there is a success message: "Success You have successfully saved MYAPIKEY." Below this, there is a table with three columns: Name, Products, and Status. The table contains one row with the following data:

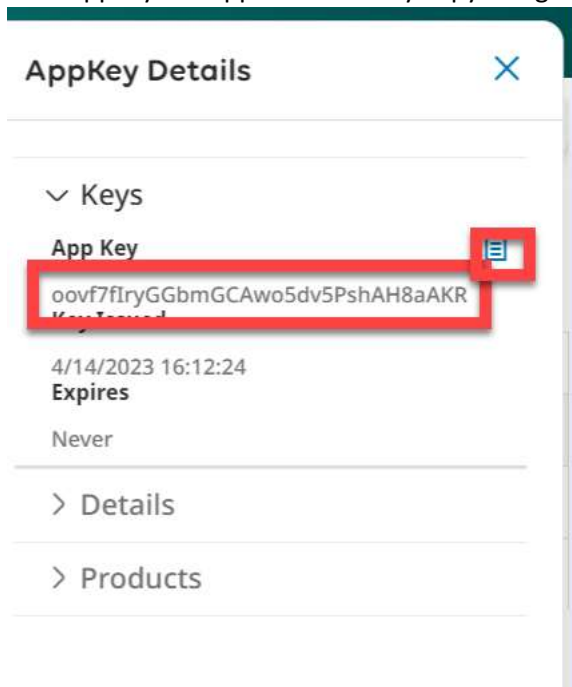
Name	Products	Status
MYAPIKEY	EDWI, FALCON_PUBLIC	approved

As I said, you don't actually use the name you entered anywhere. Highlight the key (by clicking on it) and click details at the top:



The screenshot shows the UKG My Apps interface with annotations. A red box highlights the 'Name' column, and a red arrow points to the 'Details' icon (a grid with a plus sign) in the top right corner. Another red box highlights the 'MYAPIKEY' row, and a red arrow points to the 'Details' icon in the top right corner.

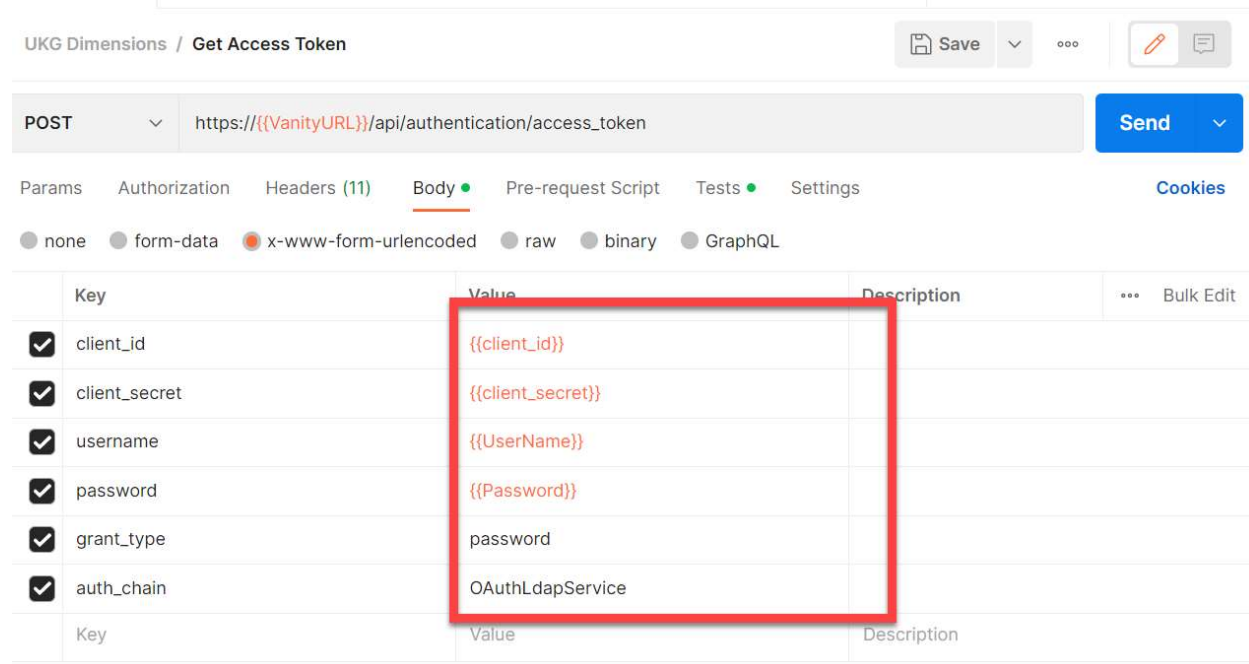
Your appkey will appear. You may copy using the icon to the top right of the key:



The screenshot shows the 'AppKey Details' modal. It displays the 'App Key' as 'oovf7fIryGGbmGCAwo5dv5PshAH8aAKR'. A red box highlights the 'App Key' text, and a red arrow points to the copy icon (a document with a plus sign) in the top right corner. Below the key, it shows the expiration date '4/14/2023 16:12:24' and the status 'Expires Never'. There are also links for 'Details' and 'Products'.

Body

The body has six values, four of which are variables.



UKG Dimensions / Get Access Token

POST [Save](#) [Send](#)

Params Authorization Headers (11) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

Key	Value	Description
<input checked="" type="checkbox"/> client_id	{{client_id}}	
<input checked="" type="checkbox"/> client_secret	{{client_secret}}	
<input checked="" type="checkbox"/> username	{{UserName}}	
<input checked="" type="checkbox"/> password	{{Password}}	
<input checked="" type="checkbox"/> grant_type	password	
<input checked="" type="checkbox"/> auth_chain	OAuthLdapService	

Client ID and Client Secret were provided at implementation. If you don't have them, you'll need to open a support ticket.

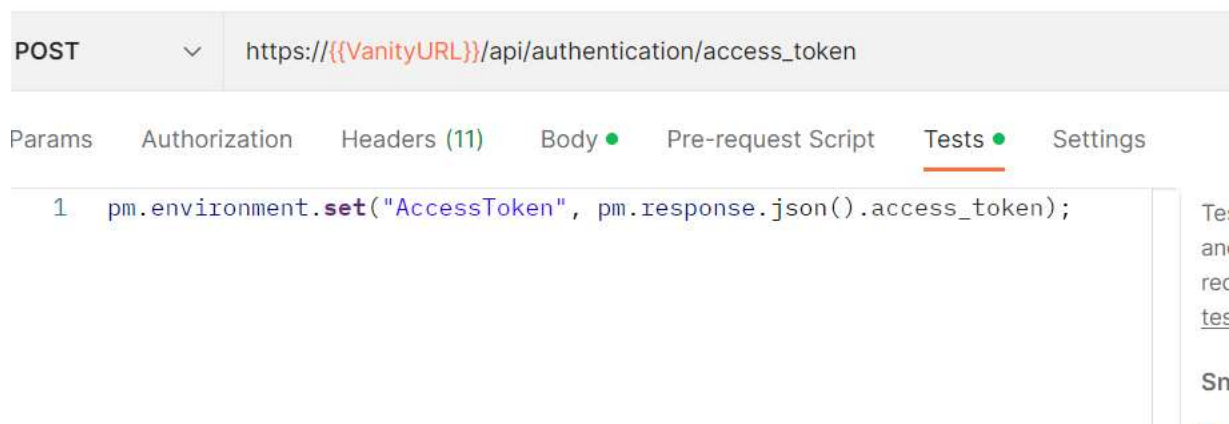
The Username and Password can be for any Dimensions user with the appropriate security.

Tests

The entire point of this request is to get the access token.

This very small piece of java script saves the accessToken into the environment variable so that it may be used in the next request.

UKG Dimensions / Get Access Token



The image shows a Postman interface for a POST request named 'Get Access Token'. The URL is `https://{{VanityURL}}/api/authentication/access_token`. The 'Tests' tab is selected, showing a single test script: `1 pm.environment.set("AccessToken", pm.response.json().access_token);`. The right sidebar shows a partial view of the 'Test Results' section.

POST `https://{{VanityURL}}/api/authentication/access_token`

Params Authorization Headers (11) Body ● Pre-request Script Tests ● Settings

```
1 pm.environment.set("AccessToken", pm.response.json().access_token);
```

Test Results

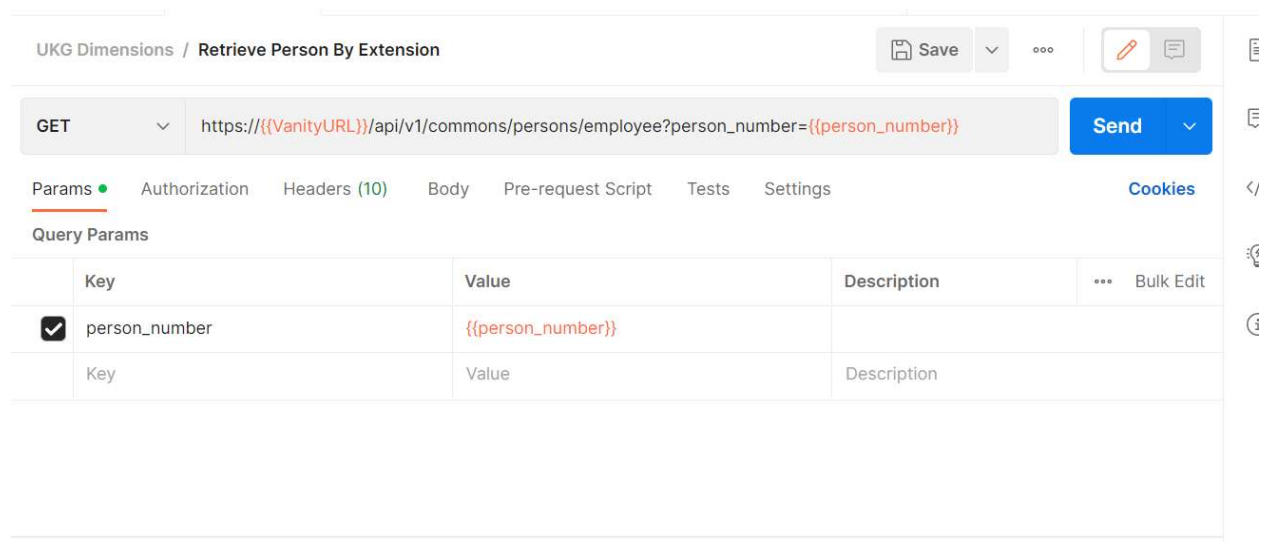
Retrieve Person by Extension

So, now you have your AccessToken. You probably want to actually do something. We're going to get information for a given employee. This is actually a simpler request. Let's parse the postman step by step.

In this case, we'll look at the request, the parameters and the Headers.

You'll see that the request uses the same VanityURL we used for the access token. Thus the great value of variables.

Also, we have one parameter called `person_number` which we assign in our environment variables.



The image shows a Postman interface for a GET request named 'Retrieve Person By Extension'. The URL is `https://{{VanityURL}}/api/v1/commons/persons/employee?person_number={{person_number}}`. The 'Params' tab is selected, showing a table of query parameters. The 'person_number' parameter is checked and has the value `{{person_number}}`. The right sidebar shows a partial view of the 'Test Results' section.

UKG Dimensions / Retrieve Person By Extension

GET `https://{{VanityURL}}/api/v1/commons/persons/employee?person_number={{person_number}}` Send

Params ● Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

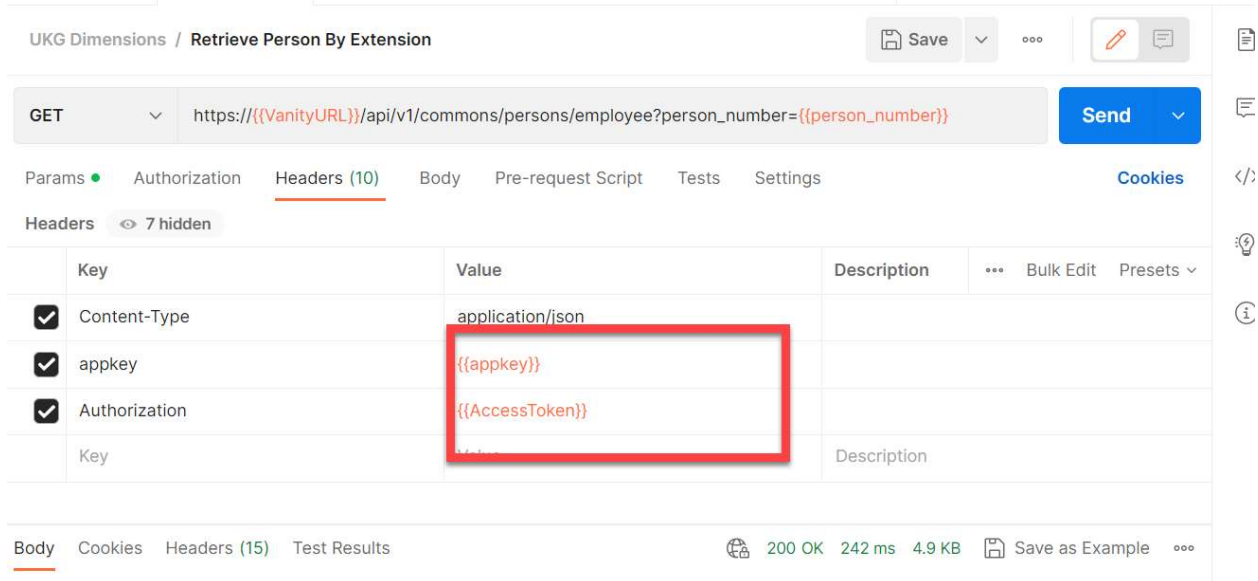
Query Params

	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	person_number	<code>{{person_number}}</code>			
	Key	Value	Description		

Headers:

The headers have three values, two of which are variables.

We've discussed both the appkey and the AccessToken previously.



Running the example.

To run the example, you need to:

- 1) Update the variables.
- 2) Run the authentication/AccessToken request.
- 3) Run the Retrieve Person by Extension Request.

(Note that we should be able to call the AccessToken Request directly from the Retrieve Person by Extension request. When I have that working, I'll update the code).

After all this, you have JSON. Which you have to parse. Which is a topic for another day.

If you have questions or comments, please leave them below. Or [contact us](#) if you're interested in training, development or consulting.

And once again, here's the link to our GitHub repository.

*They do have examples in Curl and Java. As a data guy who longs for the days when SQL was my tool, let's just say that I'm not going to learn Java. Indeed, when we need scripting, we use Powershell. And everything needs to start with Postman.