# Account Management and PebblePad API 2.0

## Introduction

PebblePad version 1902 introduces a new GraphQL based version 2.0 API to provide customers access to user account information for integration in organisational account management systems.

GraphQL provides a standardised method of interacting with APIs which allows the client to specify what data it needs in a single query rather than having to combine results from multiple endpoints.

It also allows for much more expressive queries than can be easily provided via a REST API, this allow any combination of filters to be applied as required without needing any changes to the API.

## What can you do, what are the limitations of this implementation and what is not supported?

The new API provides the same basic user information previously available in the v1.0 API, including the name, email, username and associated ids.  Beyond this, additional statistics relating to the usage of the account are available to assist in making decisions about account housekeeping tasks, these include:

- Alumni account created?
- Date the account was marked as dormant (if dormant)
- Date the account was marked for deletion (if marked for deletion)
- Date the account became active
- Date the account was created
- Last login date
- Number of logins
- Expiry date of the account (if set)
- Total size of files uploaded by user
- Number of assets created by user
- Number of assets shared by user
- Number of comments left
- Number of files uploaded

- Number of assets submitted to ATLAS

## Getting Started

To get started with the new API you will need OAuth authentication tokens. These are unchanged from the OAuth 2.0 service authentication method used by the version 1.0 API so if you are already using the existing API these will continue to work.

If you have not used the API previously you will need to contact support to request authentication tokens, please see the version 1.0 API documentation for details of how to use the tokens.

https://v3.pebblepad.co.uk/api/docs (these docs are identical across domains)

To connect with the GraphQL endpoint you will need a suitable library for your chosen development language, various options are available from the GraphQL website at: https://graphql.org/code/#graphql-clients

The GraphQL Playground developer tool provides an easy way to develop and test queries before writing any code: https://electronjs.org/apps/graphql-playground

The endpoint URLs for the production and staging servers are as follows:

| Install | URL |
|---|---|
| TAQAS | https://apptest.pebblepad.com/api/2.0/gql |
| Production – AUS | https://v3.pebblepad.com.au/api/2.0/gql |
| Production – CA | https://app.pebblepad.ca/api/2.0/gql |
| Production - Custom UK | https://capp.pebblepad.co.uk/api/2.0/gql |
| Production – UK | https://v3.pebblepad.co.uk/api/2.0/gql |
| Production – US | https://pebblepad.com/api/2.0/gql |

# Connection and First Query Walkthrough

## Step 1

Request OAuth 2.0 service tokens with permission to read user accounts from support@pebblepad.com

## Step 2

Download and install GraphQL Playground from https://electronjs.org/apps/graphql-playground

## Step 3

Launch GraphQL Playground and fill in the GraphQL Endpoint field, enter the URL for region as listed above and open.

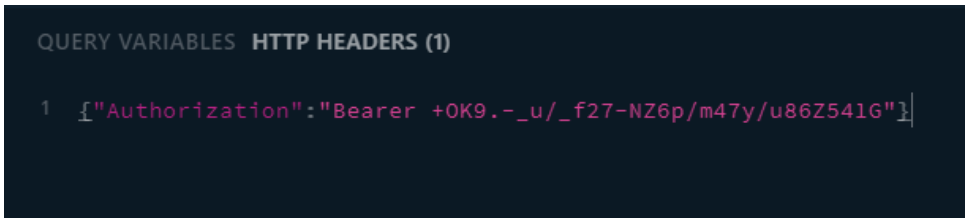
This example will use the TAQAS staging server which you can also request access to if you wish to test your integration before taking it into production.

We recommend testing on TAQAS if you intend to develop and integration that make automated changes, such as one that automatically removes user accounts.

## Step 4

On first connect you should expect to see a 401 error message due to the OAuth tokens not being set.


To provide these details, use the HTTP HEADERS tab in the bottom left to enter the Authorization header.




This needs to be entered in the following format, replacing '<Your OAuth access token>' with the access token provided to you by support as the value. Bearer at the start of the value is required and must be kept.

**Edit HTTP Headers**

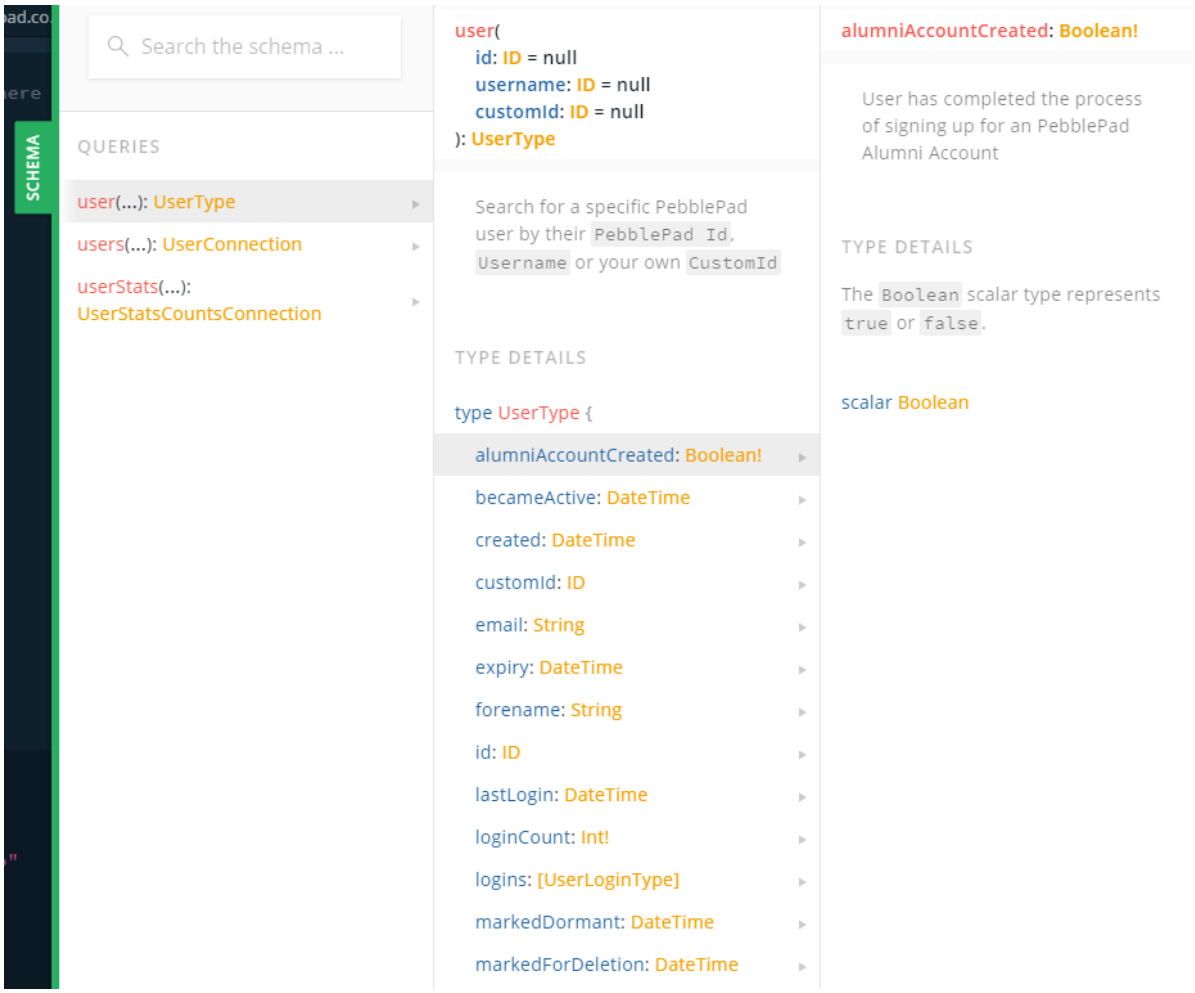
| Header name | Header value |
|---|---|
| Authorization | Bearer <Your OAuth access token> |

Step 5

As soon as you finish entering the authorization header the error message should disappear. You should also be able to open the Schema tab on the right-hand side, this provides details about all the available fields you can query and retrieve.

If the error does not clear you may need to refresh your access token, these only remain active for 60 minutes on production servers for security reasons, please see the “Refreshing your Access Token“ section in the main API getting started for details:

https://v3.pebblepad.co.uk/api/docs



## Step 6

To run your first query paste the following into the query window to fetch 10 users who have never logged in.

```
{
  users(first: 10, loginCount: "0") {
    items {
      username
      forename
      surname
      lastLogin
    }
  }
}
```

In this example “users” represents the query to run, within the parenthesis are the parameters passed to the query.

Lists returned by the API are paginated using the Cursor Connections Specification, as a convenience function we also provide the results of the query in items, adjacent to items you can access information provided by Cursor Connections regarding the availability of further data.

Within items you specify the fields you are interested in returning, unlike REST APIs GraphQL only returns the information you request, reducing network traffic and improving performance.

Please note that the GQL endpoint applies a rate limit to GET requests made to it, the current limit is 100 per minute.

# Examples

Some examples of useful queries

Find users that have logged in between two dates and show how many assets they have along with their engagement count

```
{
  users(first:100 lastLogin:"2018-11-25T23:41:08.013Z..2018-11-29T23:41:08.013Z") {
    items {
      username
      lastLogin
      statsCounts {
        assetCount
        engagementCount
      }
    }
    totalCount
  }
}
```

Find users that haven't logged in since a specific date that also have created an alumni account

```
{
  users(first:100 lastLogin:"<2018-05-31T23:41:08.013Z" alumniAccountCreated: true) {
    items {
      id
    }
    totalCount
  }
}
```

Find users that don't have any assets

```
{
  userStats(first: 100 assetCount:"0") {
    items {
      userId
    }
  }
}
```

Get the number of assets, shares, files and comments for a specific set of users


## Further Reading

GraphQL Tutorial

https://www.howtographql.com/

Cursor Connections Specification

https://facebook.github.io/relay/graphql/connections.htm