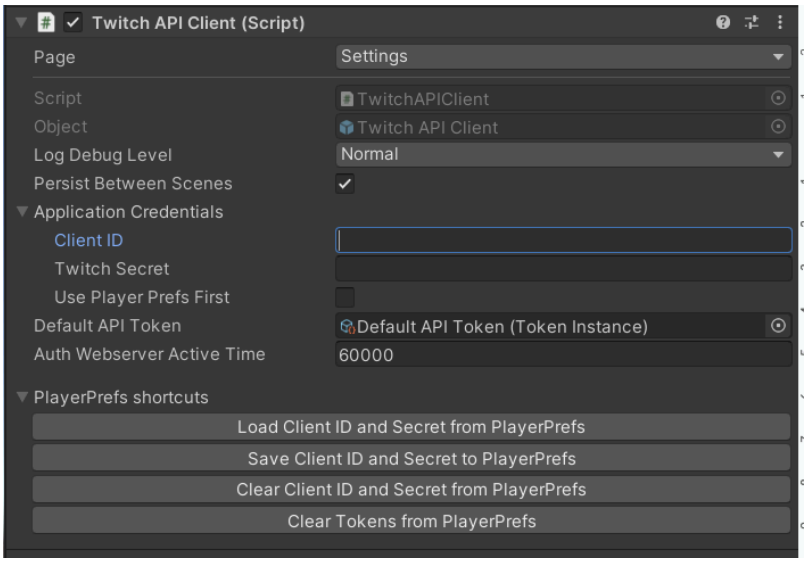
Getting Started with StreamLinked

Created with reference to version 2.2.0

StreamLinked is a collection of GameObjects and classes to allow a user to quickly and easily call Twitchs API from inside the Unity game engine.

Initial setup



The centre of the functionality of StreamLinked revolves around the singleton class TwitchAPIClient. This class manages web calls to Twitch and maintains the authentication used for the requests.

To make calls to Twitch this object must be present in the scene. A gameobject is provided with this script attached in prefabs.

For basic setup and making your first call to twitch it is recommended not to change any settings from the current set defaults.
The only value that needs to be provided for this example is the Client ID. With this you will be able to make an Authorisation call using Implicit Grant Flow.
To get this value you need to set up an application on Twitchs developer console here.

With this done, play the scene and the API will get you a new token and store it in Unitys PlayerPrefs.

Making a call to the API

With the client in the scene and authorised, you can now make calls to Twitch API. When making calls to Twitch API please be mindful of values you provide and how many times you make requests, Twitchs documentation on this can be… questionable in places.

Here I will demonstrate how to make a call to Twitchs API. I will translate Twitchs documentation to the method in question in the next section.
A scene has been pre included called API test scene. Open this scene for an explanation of how it works. On the canvas object, there is a script called GetUserTestScript. Opening this

script in your code editor you will see how to make a call using the enumerator version of the request (And other types commented out underneath).
On line 38 you will see the method to create a request to Twitchs Get Users.
[TwitchAPIClient.MakeTwitchAPIRequest<GetUsers>(...)]

Get users is a basic call to Twitchs API to get a provided users information, it is really handy if you need a users ID instead of their login name for example. Get users can be called without any provided parameters as well but for this example we are providing the name in the input box above.

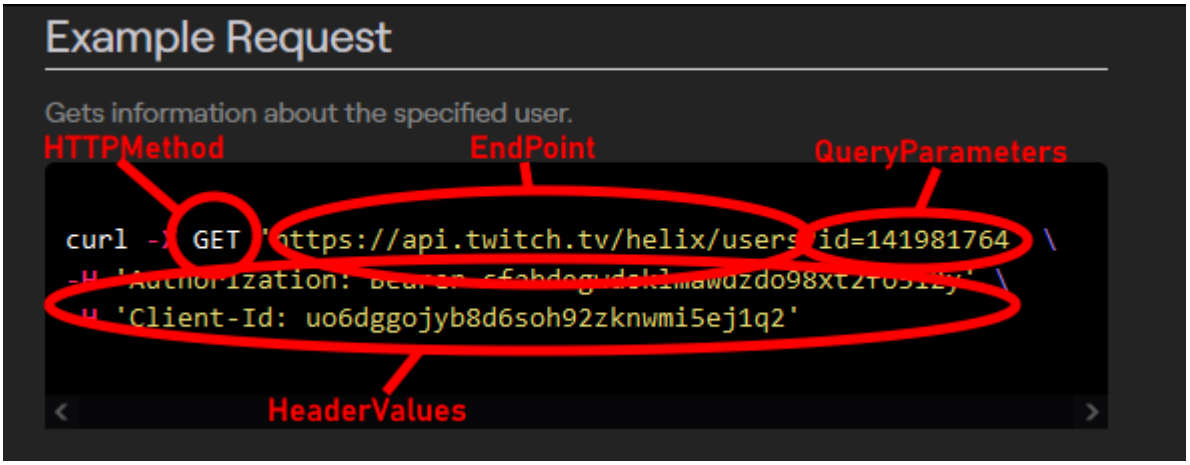
The second part of this request with coroutines (if you care about the response) is the callback. Coroutines don't have a natural way of returning a value so I have provided a callback function with the returned data to use it.
(There is also Async version of the request that I highly recommend you use over coroutines, they dont require callbacks and can just be awaited for the value, there is a synchronous version too for testing purposes however this is not recommended for development builds)

TwitchAPIDataContainer

The returned value from the request is packaged inside an object called a TwitchAPIDataContainer<T>. This container stores all possible information returned from a Twitch API request. Depending on the method you call a lot of fields will probably be empty as it stores information on if the method errored, error information, date range, pagination etc.
The information we care about on a successful request is the data array. This array is all the returned information from the request, many methods return multiple rows although if you are only expecting 1, it's common to default to getting the information using (data.[0]).

You can see this being applied in the SuccessCallback function found at the bottom of the script that was opened.

Translating dev.twitch.tv/docs/api/reference to MakeTwitchAPIRequest



Get Users

With this information you can now make requests to any endpoint on Twitchs reference page. Each endpoint has its own class provided with all of the information to make a request. The description of each class contains a link to the corresponding reference point on Twitchs website, all returned values are available as parameters, HTTPMethod and Endpoint are provided as Gets in each class and lastly each available values that can be provided as values to the request are as static strings in capital letters (e.g ID and LOGIN in GetUsers).

An override is provided to auto populate the classes HTTPMethod and Endpoint into the request if a generic of the return type is provided. **The API method will automatically inject the 'Authorization' and 'Client-Id' as well.** Everything else you will need to provide if it is required.

Some endpoints will require a special field called Body. This field represents the Request Body on the reference page. If the method supports it, a static method is provided called BuildDataJson to build the data into a json string to provide into the method, the parameter name is Body.

If you didn't know already with C#, with methods any provided variable that is preassigned a value (so you don't need to fill it in) can be quickly applied a value without needing correct positioning by specifying in a space the name of value you wish to populate.
For example if you want to add the Body value without adding the other parameters. You can write it as; MakeTwitchAPIRequest(.. , Body: "Your value here");

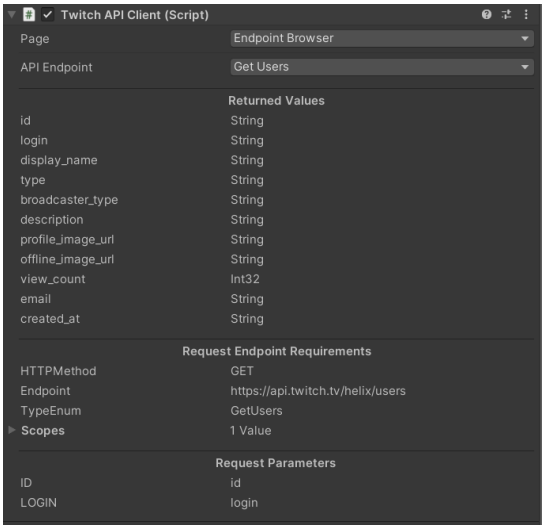
More Tokens?

StreamLinked comes with 3 precreated tokens for you to use, one for the API, IRC and EventSub. Needing one for each isn't required and can be operated with a single token provided to the API client, if tokens are missing from the IRC and EventSub requests default to the API client to make the request.
To make a new token, go to StreamLinked menu bar and select 'Create OAuth Token' and it will make a new Token in the base folder for StreamLinked (/Assets/StreamLinked/)
Alternatively you can right click, Create, StreamLinked, Create Token and it will create one in the folder you have right clicked in.

Endpoints and Quick Requests

You can now browse API endpoints inside the inspector. Found on the page 'Endpoint Browser' you can now search API Endpoints and it will return for you all the known information associated with it.
Starting at the top it displays the Returned Values, on the left is the known name of the value and on the right is the type of value it is. Arrays are indicated with the traditional braces[] and sub objects are rendered as dropdowns listing their own parameters.

Next is the Request Endpoint Requirements, these are values needed to make the Twitch request such as the HTTP method and Endpoint. Also included is the Enum value associated with it and scopes (if any) are associated with it.
Lastly is the Request Parameters, these are the names of possible values that can be sent up with the request, on the left is the static name that can be accessed from the Type, for example; GetUsers.LOGIN. On the right side is the constant text associated with it and is sent up to Twitch. Also in this section is a field for if a Json body needs to be sent up with the request.



Lastly is a page to learn more about your Token Instance.
Found on the last page of the API client called ‘Quick Requests’. This page will ask for a Token Instance from you, after doing so you will be able to ready any token information stored for it. If no token is provided, you can from this page attempt to make an OAuth request for this token. Once a token is known, you will be able to from this page make requests to Twitchs API using any endpoint provided. **This page is very experimental and StreamLinked was not created with this functionality in mind, it is very fragile and is intended for testing purposes.**

Results are printed inside a text area created when a response is returned in a pretty format to understand the results returned. Using this you can make sure the endpoints you are calling are successful without requiring play mode to test them.

Do take caution with using this, not all endpoints are going to work correctly with it and being Twitchs API some may require immediate follow up that is not supported here. Consult the documentation before making actual Twitch based changes.