Dynamic Advertisement Documentation

Unreal Engine 4

v1.0.6

**Overview**

The dynamic advertisement platform allows brand agencies to control and monitor their campaign activities directly with your game, while giving you full control of what campaigns go live.

In order for this service to be provided you will need to integrate with our technology. We have built this API service to be as straight forward as possible to reduce the level of development work required.

The advantage is that once integrated, we will be able to run multiple campaigns with many different brands, without your game requiring patches or updates for your players and increases the revenue potential dramatically.

Useful Resources

**REST API Docs**

Our official documentation for direct API integration

<https://dynamicadverts.docs.apiary.io/>

**VaRest Plugin**

If you decide to do a direct integration, you can utilise the VaRest plugin to help with requests to the backend server.

[**https://github.com/ufna/VaRest**](https://github.com/ufna/VaRest)

**Direct Integration**

Direct integration allows you to connect to the API in whatever way you see fit. This may be your preferred method of integration if you wish to have greater control of how it works with your game.

**API Endpoint:**

**https://hub.raidparty.io/sdk/advert/**

**Process:**

1. **New Player Session Starts**

Once a player connects and loads the game, send a request with the following payload of data:

* 1. Game ID (provided to you)
  2. Unique Player ID (your own unique player ID)
  3. Player IP address e.g. 37.122.196.70

The server will respond with either a HTTP 200 success or HTTP error code as defined in error codes.

Additionally it will return player location information for your own records should you have a use for it.

1. **Find an advert campaign for each game object**

For each game object that is available for advertisement placement, send a request with the following payload of data:

* 1. Game ID (provided to you)
  2. Unique Player ID (your own unique player ID)
  3. Game Object ID (provided to you for each game object)

The server will respond with a payload of advertisement assets (such as video URI or texture(s) to assign to the game object or an empty success response if no adverts are available.

Best practice would suggest you send these requests as early as possible and store the advert assets on the client device so they can be rendered prior to the game starting. You should also have default textures in place for cases when no advert campaigns are available for that particular player/game.

1. **End Player Session**

It is important this final request is made so that accurate reporting can be provided to our clients. However, in some situations (such as player game crashes) this will not be possible. Our system will automatically adjust based on previous behaviours.

At the end of the players game session, send a request with the following payload of data:

* 1. Game ID (provided to you)
  2. Unique Player ID (your own unique player ID)

The server will respond with either a HTTP 200 success or HTTP error code as defined in error codes.