Upgrading to brainCloud from PlayFab

Matchmaking

Table of Contents

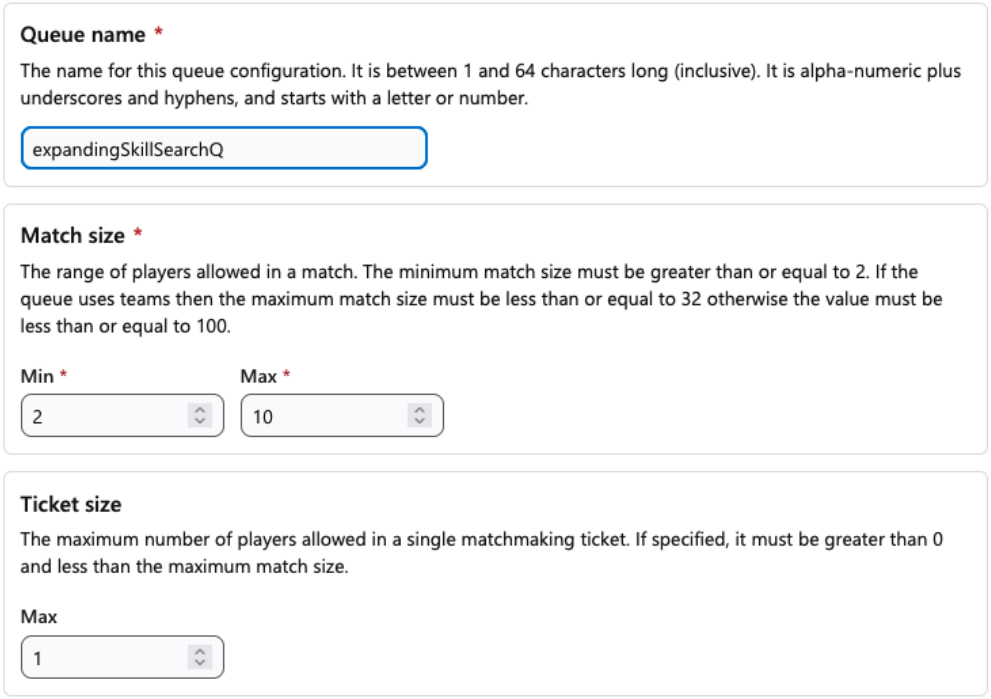
Introduction 3
Matchmaking Queues 3
Configuring brainCloud Lobbies 5
Enabling Lobbies 6
Matchmaking Flow & Portal Testing 9
Enabling RTT 9
API Explorer 10
FindOrCreateLobby Flow 11
Complex Rules & Filter Scripts 15
Portal Testing 17
PlayFab Lobbies 20
Lobby Invites 21
Offline Matchmaking 21

Introduction

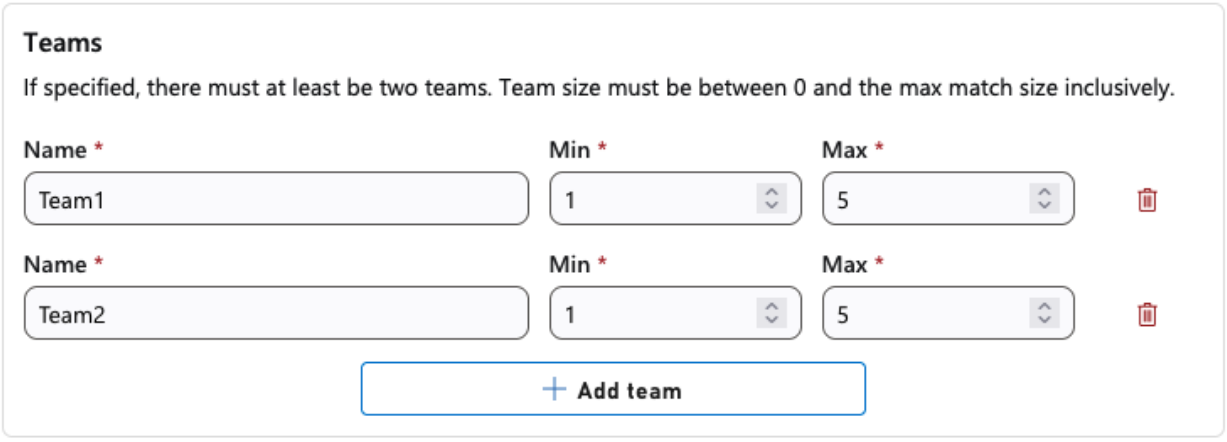
PlayFab and brainCloud both provide matchmaking solutions which also integrate with their hosted servers for real-time networking. However, their approaches differ. This guide outlines how brainCloud's matchmaking features correspond to those in PlayFab and highlights additional offerings from brainCloud that can facilitate migration or enable future feature development.

Matchmaking Queues

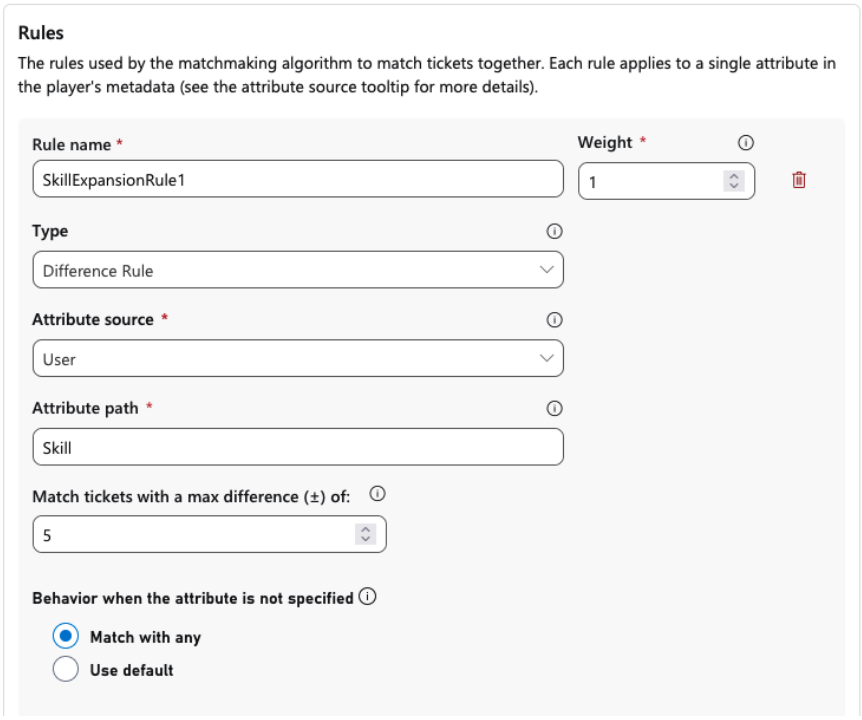
Let's begin with PlayFab's most basic form of matchmaking: a simple queue with expansion rules.



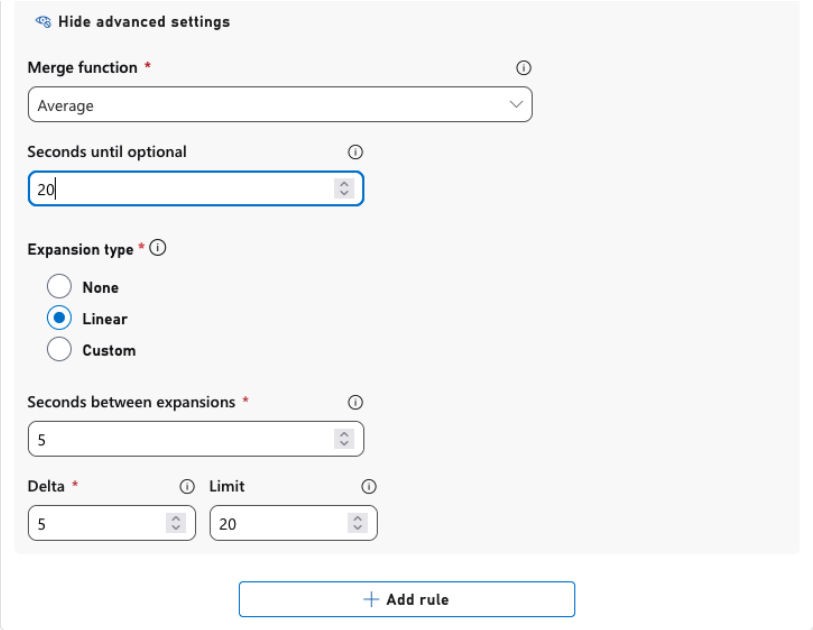
PlayFab offers team setup options within matches. This guide will also explore how to replicate team configurations using brainCloud.



Lastly, we'll add expansion rules that can be replicated in brainCloud. Players will submit a "skill" value for matchmaking. The matchmaker will attempt to match with other players with the same skill +/- 5 and overtime we will increase the range for the skill.

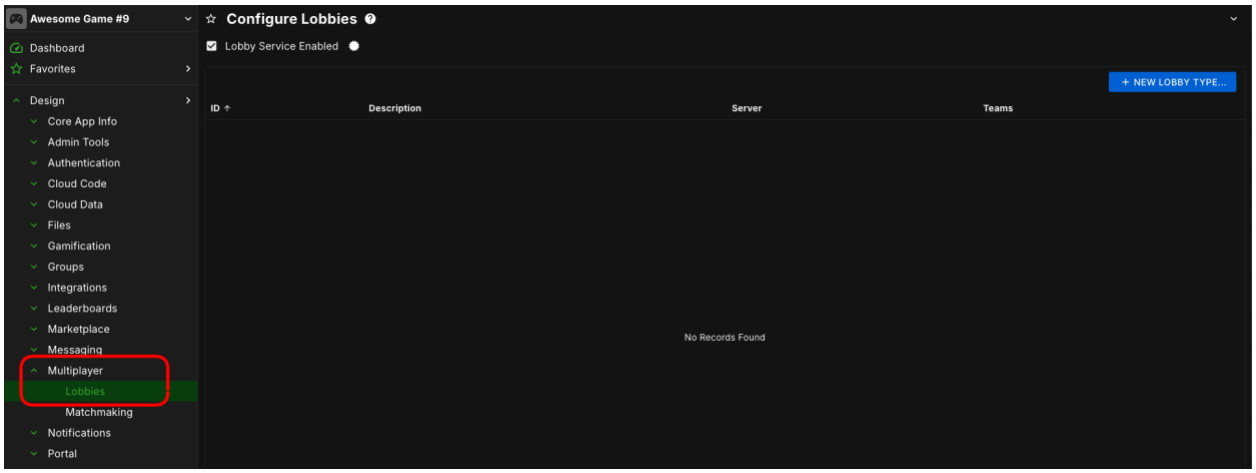


Every 5 seconds it will expand this range by 5, up to a maximum of +/- 20. After 20 seconds it will expand to include a match with any skill value.



Configuring brainCloud Lobbies

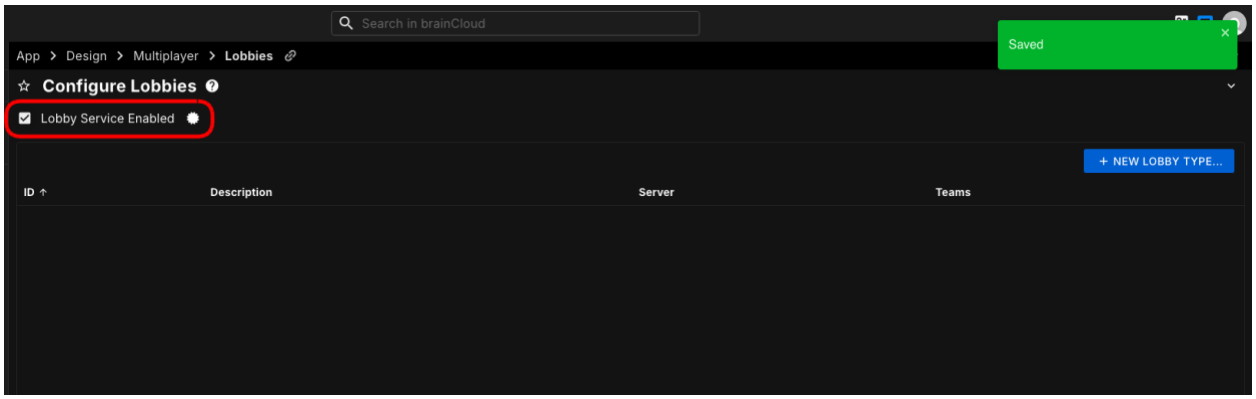
Now, let's replicate this matchmaking configuration with brainCloud. These settings are configured through the portal. Navigate to the **Design → Multiplayer** menu to access them.



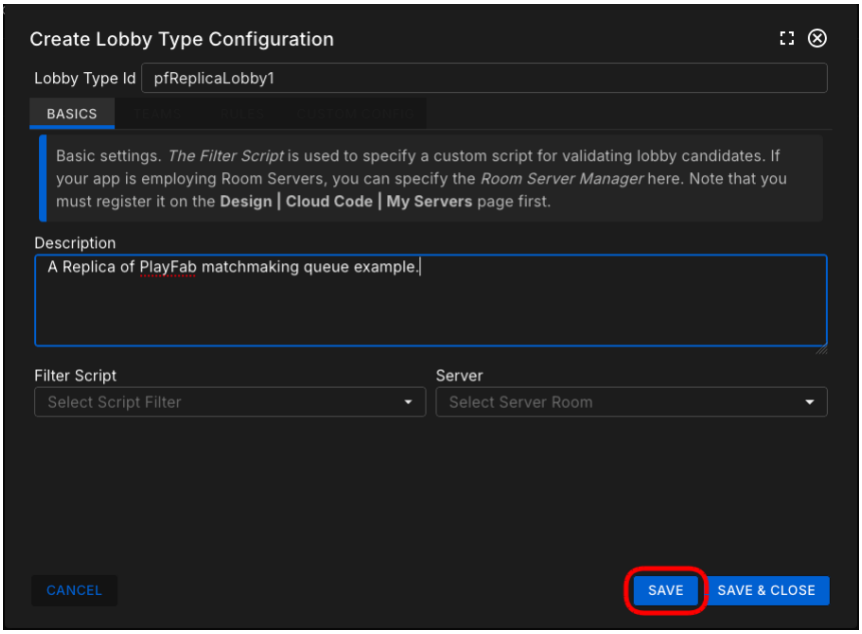
Here we are selecting Lobbies. This might be confusing for those familiar with PlayFab, which also has a concept of Lobbies. We'll later compare the two, as they're quite similar. For now, consider brainCloud's Lobbies analogous to PlayFab's matchmaking queues.

Enabling Lobbies

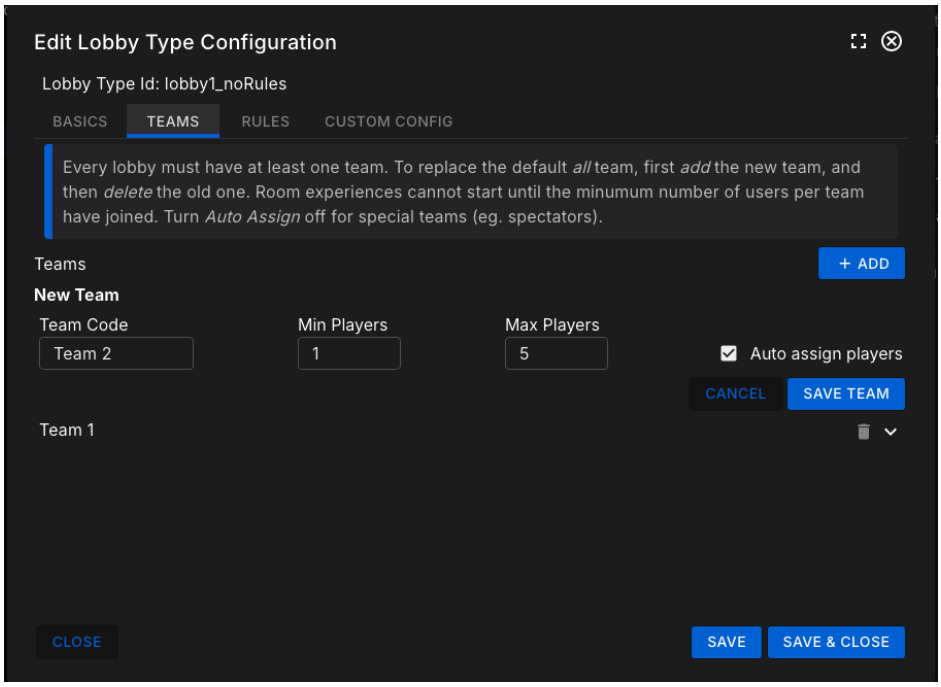
The first thing we need to do is enable Lobbies for your brainCloud application. You can do this from the menu screen.



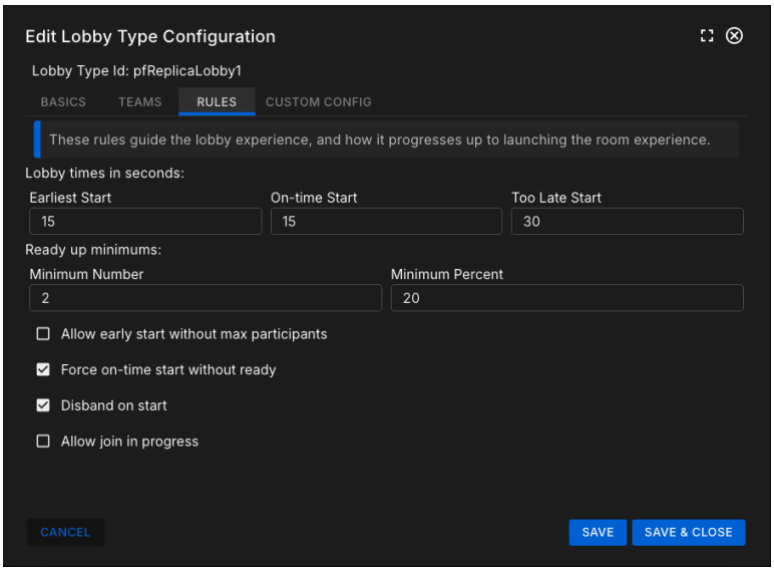
Click on the "+ New Lobby Type" button to configure a new lobby.



For now, assign the lobby an ID and a description. You don't need to worry about the *Filter Script* or *Server* options at this stage; we'll cover those later in this guide. Click *Save* to reveal new tabs, and then click on the *Teams* tab to explore the options available there.



Here, you can set up the same team configuration as with your PlayFab queue. Note that you'll need to delete the default team called "all", however, If you don't want to use teams, this "all" team functions like your min/max players setting. Click **Save** to apply these changes. Next, go to the Rules tab to modify some rules.



We'll configure this lobby to mimic a simple PlayFab matchmaking queue. Let's examine some parameters and their relation to PlayFab's queues:

- **Earliest / On-time / Too Late Start**
  In PlayFab, this is similar to the `GiveUpAfterSeconds` used in the `CreateMatchmakingTicketRequest`. A "Too Late Start" value of 30 means if a player hasn't joined an ongoing lobby, they create a new one and wait up to 30 seconds before giving up. We'll explore this later with examples.

- **Minimum Number/Percentage**
  These represent the minimum players needed to start. In PlayFab, we set the minimum match size to 2. Here, set the *Minimum Number* to 2 and *Minimum Percent* to 20. With a team size of 5 and 2 teams, the maximum in this lobby is 10, so 20% is 2 players. The lobby can start when both these requirements are met.

- **Allow Early Start Without Max Participants**
  This lets the lobby move to STARTING once the minimum is reached. Disable this to prevent forming the lobby before having enough players.

- **Force On-Time Start Without Ready**
  A handy feature in brainCloud not found in PlayFab; players can mark themselves "ready" which will force the lobby to check all players are ready before progressing to the final state. Since PlayFab lacks this, enabling this allows lobbies to start regardless of player readiness.

- **Disband On Start**
  Enables deletion of the lobby once a match is found. brainCloud provides needed info asynchronously during the match, so retaining the lobby post-match is usually unnecessary. If using PlayFab match data for other purposes (like PlayFab lobbies), consider disabling this.

- **Allow Join In Progress**
  This is for "backfilling" lobbies which are already in an active and generally used for hosted room servers so we don't need this enabled for this example.

These rules for brainCloud's matches can interact to create more complex behaviours depending on your requirements so you can learn more about these rules here. Once configured, click the **Save & Close** button to finish this setup.

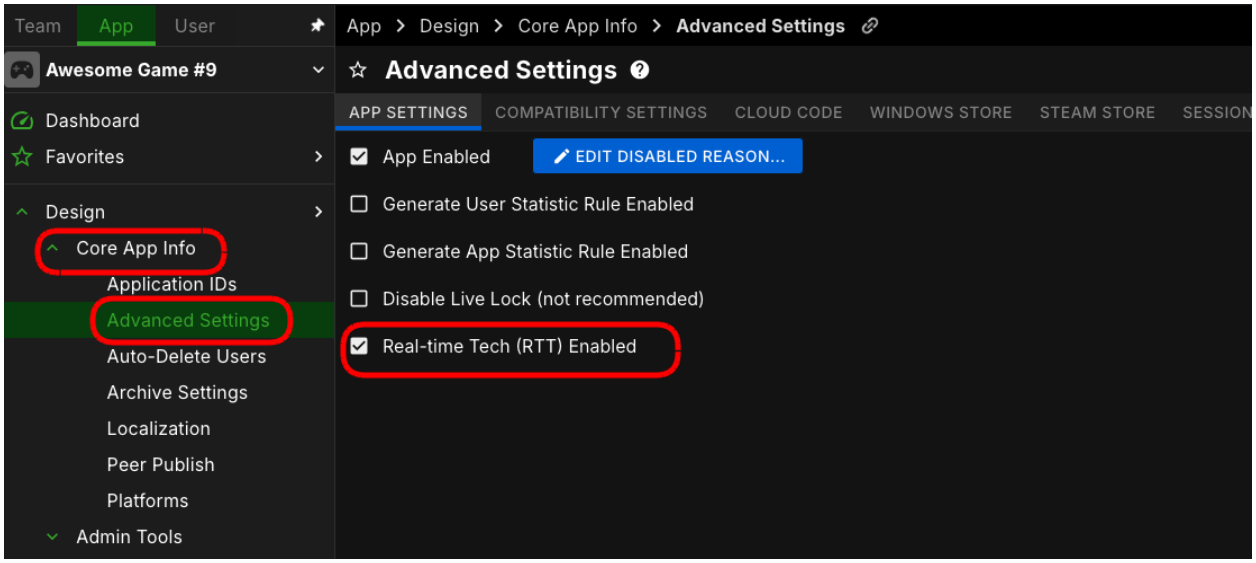
Matchmaking Flow & Portal Testing

Next we'll want to test the matchmaking flow to ensure the same behaviour has been replicated with brainCloud.

One important difference between brainCloud's lobbies and PlayFab's matchmaking is that brainCloud uses Real Time Technology (RTT) for asynchronous messaging eliminating the need for polling to receive matchmaking updates. This feature extends to other brainCloud functionalities and allows developers to create their own asynchronous APIs.

Enabling RTT

brainCloud allows testing RTT messaging via the portal's API Explorer, so you can test without first building these APIs into your client. Let's do that by first enabling RTT for your brainCloud application by going to Design -> Core App Info -> Advanced Settings.

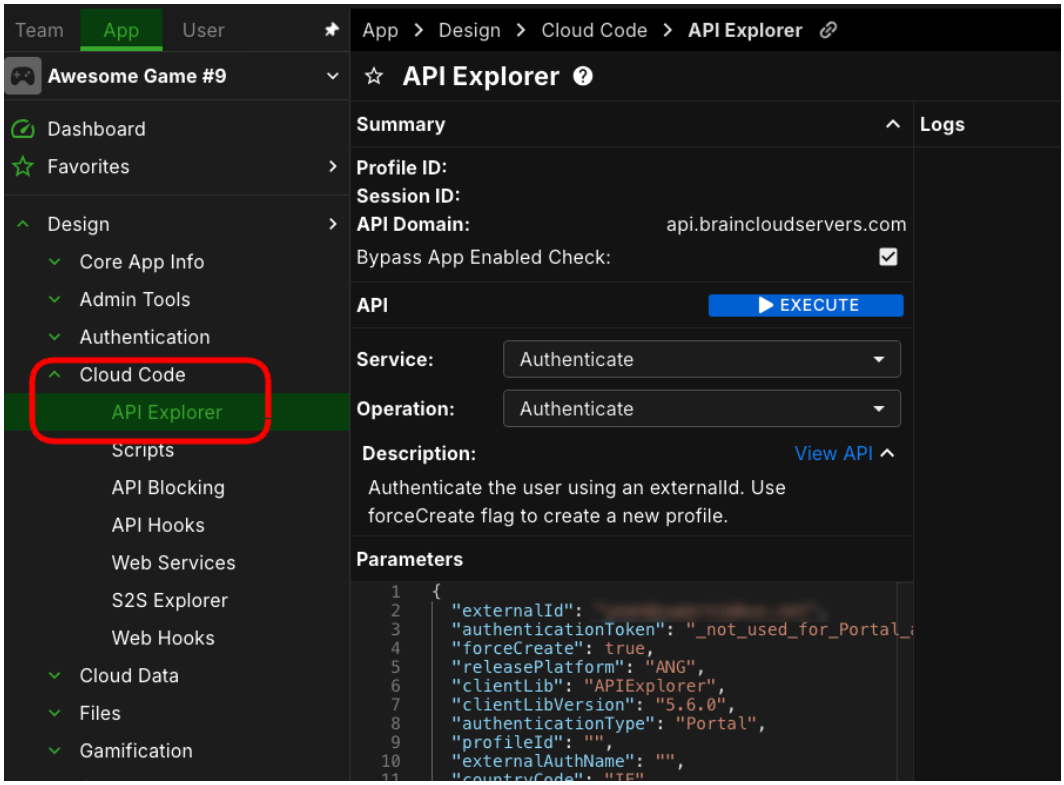


Note
Enabling RTT for your application may come with additional costs. You can learn more about plans & pricing here.

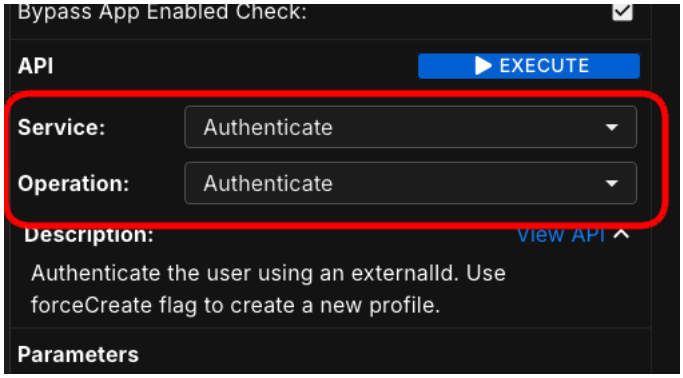
Now that RTT is enabled we can start testing matchmaking & lobby requests from the API Explorer.

API Explorer

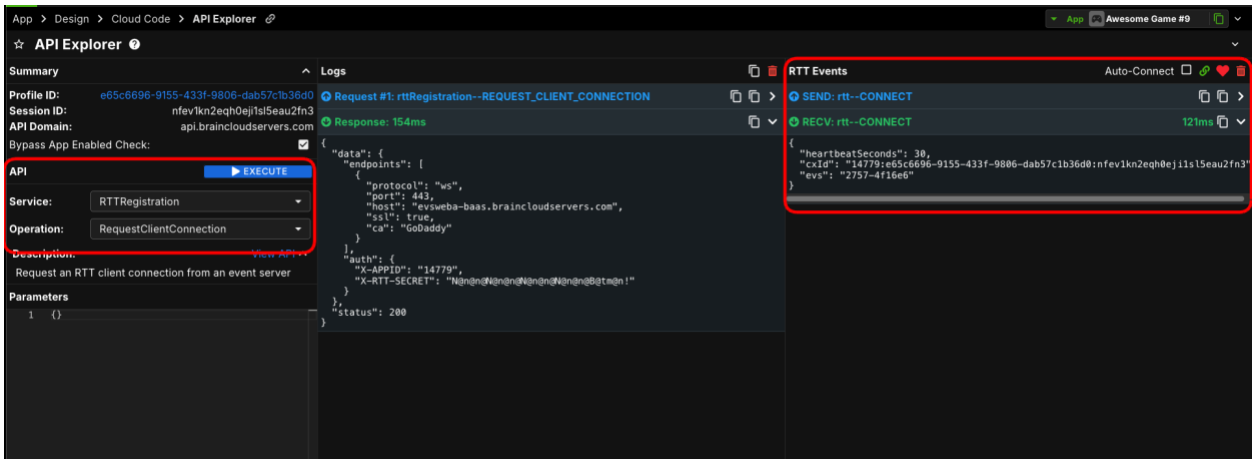
We can test our APIs using the API Explorer. You can find the API Explorer in the **Design → Cloud Code** menu.



Here, you can execute all client APIs and some system APIs that aren't callable from the client. On first opening this dashboard, you'll see the *Service* and *Operation* dropdown as **Authenticate**/**Authenticate**. Click the Execute button to log into your app as the portal user.



Next, request an RTT client connection. Select **RTTRegistration** for the API service and **RequestClientConnection** for the *Operation*. Execute the request, and you'll see a new "RTT Events" bar appear on the right-hand side.



This bar will now show us all the asynchronous messages coming in from the backend for this user.

FindOrCreateLobby Flow

With RTT messages working, let's explore the equivalent of submitting matchmaking tickets for players.

We'll use the FindOrCreateLobby API, which mirrors PlayFab by attempting to find an eligible lobby and creating one if none are available. brainCloud also offers APIs for creating, listing, and updating lobbies, but we'll focus on FindOrCreateLobby for this scenario.

In the API Explorer, select **Lobby** as the *Service* and **FindOrCreateLobby** as the *Operation*. Optional parameters will appear, but focus on adding the required ones as defaults.

```
{
  "lobbyType": "pfReplicaLobby1",
  "rating": 100,
  "maxSteps": 4,
  "algo": {
    "strategy": "ranged-absolute",
    "alignment": "center",
    "ranges": [
      5,
```

```
      10,
      15,
      20,
      100
    ]
  },
  "isReady": true,
  "extraJson": {},
  "settings": {}
}
```

Let's take look at what these parameters are doing. We'll ignore the ones that aren't important.

- **lobbyType**
  This is the name of the lobby type we created previously.

- **rating**
  This value is what we're matching. In PlayFab, this is an attribute in the `CreateMatchmakingTicketRequest` and can have any name. In brainCloud, we use the "rating" parameter.

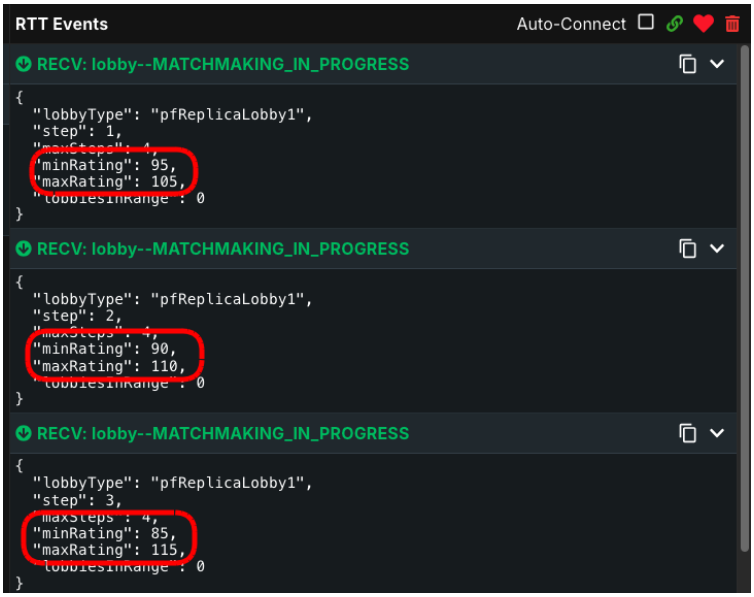
- **maxSteps**
  This is the number of steps we wait for when looking for a match. Each step will expand the range of the search. Each step represents 5 seconds.

- **algo**
  This is the search algorithm for expanding the search in steps. We use the "ranged-absolute" strategy, similar to PlayFab's *Difference Rule* type. For alignment, choose "center" to align with +/- the submitted rating. The ranges replicate the automatic increments in PlayFab but are manually set for brainCloud, with each range corresponding to a matchmaking step.

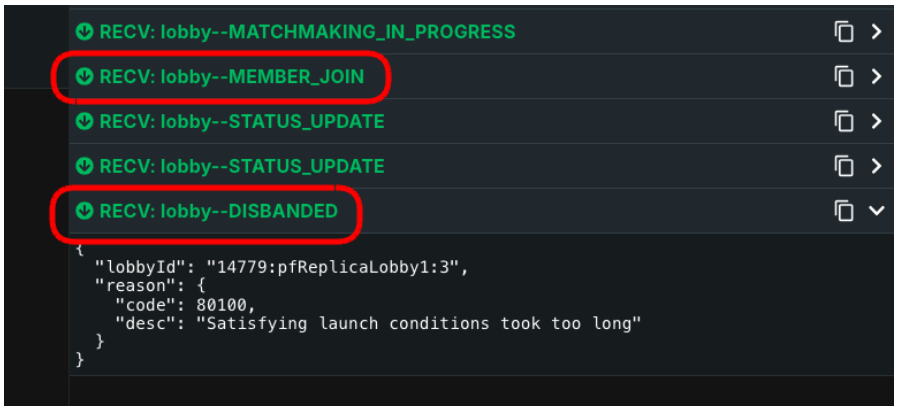
To summarize, we'll request to find a lobby with players at a rating of 100. Every 5 seconds, the range expands by +/- 5. After 20 seconds, it expands to +/- 100, ultimately including all players in our example.

Add these parameters to the request and execute the request so we can test this out.

After a few seconds you will start to see messages coming in from the RTT Events bar. You'll see the contents of the first ones are showing us how the range is being expanded.



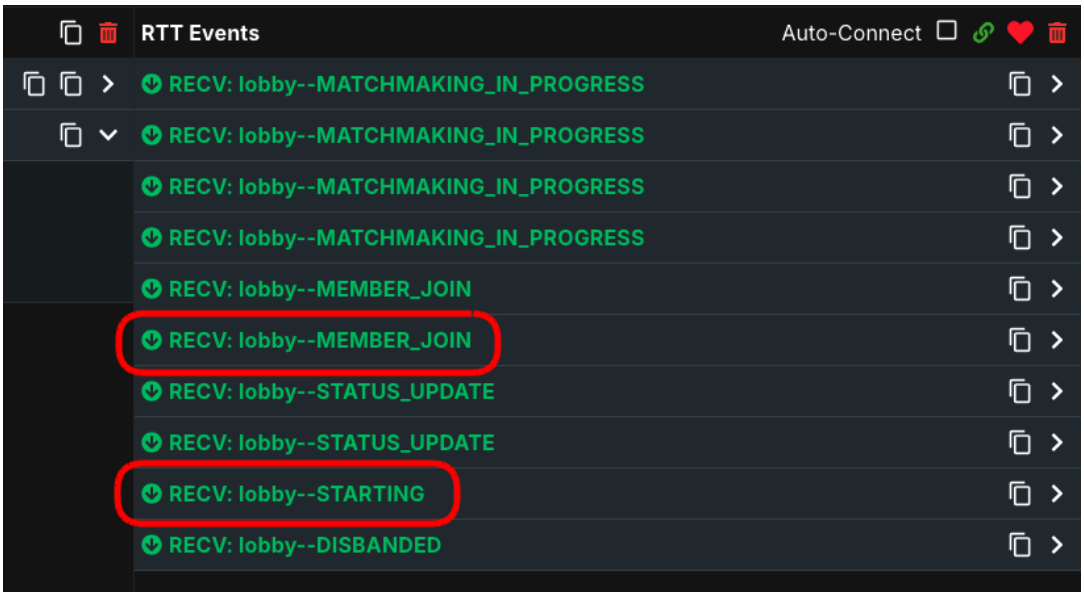
After 25 seconds, you'll see the MEMBER_JOIN message, indicating the player couldn't find an ongoing match and created a new lobby instead. Later, another message will show the lobby was disbanded because it couldn't start within the 30-second timeout due to insufficient members.



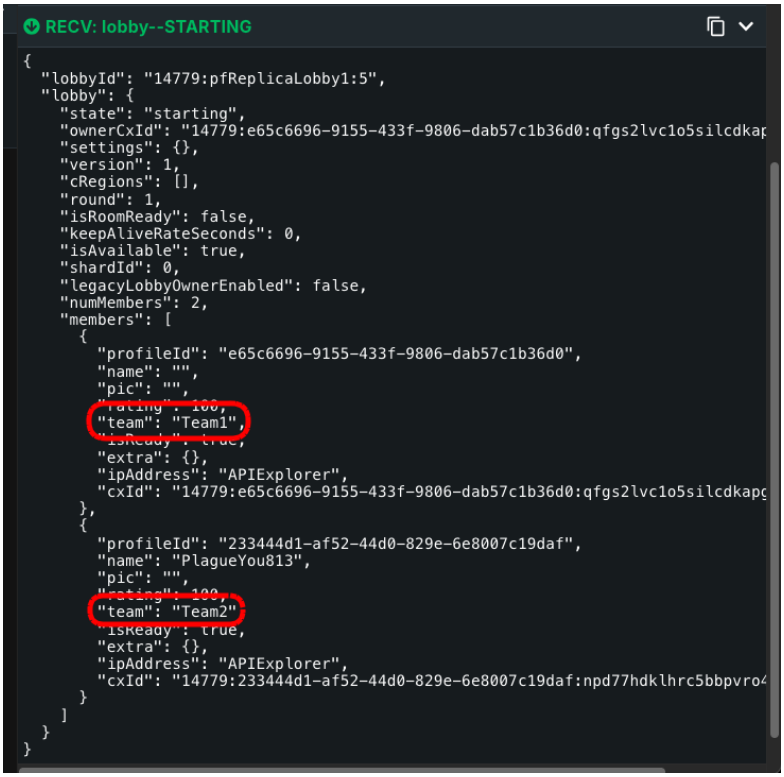
The flow starts by attempting to find an existing lobby, expanding the range every 5 seconds for 25 seconds. If none are found, it creates a new lobby to allow others to join for an additional 30 seconds before disbanding.

This isn't exactly the same as PlayFab's matchmaking but illustrates how it can be replicated. There are also APIs for CreateLobby and FindLobby for more customized matchmaking flows.

To test with other players via the API Explorer, find a player in the Users menu and log in as that player. Register the RTT callback and send the FindOrCreate API request again. You should send the request using the original player first, then wait 5-10 seconds before the second player's sends the request This will ensure a lobby is available for the second player to find.



In this scenario, you'll see two MEMBER_JOIN messages. One for each player, indicating both joined the match. With two players, the minimum members required to start the lobby was met, so the lobby moves to the STARTING state. The messages will also confirm each player is assigned to a different team as expected.



In our example, we didn't reach the 10-player requirement (as we only tested with 2 players), so the lobby was disbanded after timing out. However, before disbanding, we received messages with member details and IDs, which can be used to transition players to other multiplayer scenarios you might need for your own migration.

Complex Rules & Filter Scripts

With PlayFab, you can create complex matchmaking rules using custom parameters, statistics, or player data. For example, ensuring players in a multiplayer session own the required DLC before joining a match using DLC maps. To prevent hacking, it's safer to have these checks on the backend.

Let's explore how to achieve this using brainCloud filter scripts. We'll use a filter script to check if the player owns a Virtual Good corresponding to a DLC code. First we create a new script called “lobbyDLCFilterScript” (For detailed instructions on Cloud Code creation and setup, please refer to the Cloud Scripts Guide.

Here's a simple script with three steps:

1. **Check for DLCCode**: Verify if a DLCCode parameter is passed into the lobby settings.

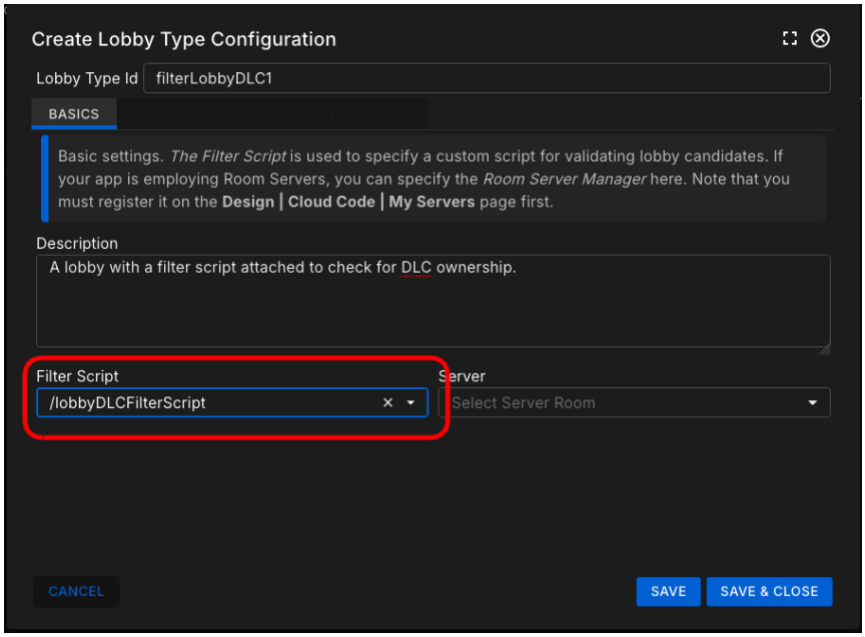
2. **Verify Ownership**: Check if the player owns a Virtual Good corresponding to the DLCCode.

3. **Set Acceptance**: If the player owns the DLC, set the "accept" parameter to "true".

```
"use strict";


function main() {
 let response = {};
 let isAccepable = false;


 const matchCandidate = data.matchCandidate;
 const lobbyDLCCode = matchCandidate.settings.DLCCode;


 // [1] - If we are looking at a lobby that has DLC settings we have to check  //
 //         the player has the VG, otherwise they are fine                      //
 if(lobbyDLCCode != undefined){
   // [2] - Check this player has the VG //
   const postResult = bridge.getUserItemsServiceProxy().getUserItem(lobbyDLCCode, false);
   if (postResult.status == 200) {
     // [3] - Set the player to be acceptable for this lobby //
     isAccepable = true;
   }
 }
 else {
```

```
    isAccepable = true;
  }


  response.accept = isAccepable;
  return response;
}


main();
```

Save the script and return to the lobby configuration menu. Create a new lobby called `filterLobbyDLC1`. Set up the teams and rules as needed, but ensure you select the *Filter Script* you just created.



Portal Testing

As in the previous example, test this lobby using the API Explorer with two players. Here's the expected flow:

1. The first player creates a new lobby after failing to find an existing one.
2. The second player meets the matchmaking criteria.
3. The filter script checks for the required Virtual Good the second player needs to join.

Note
This example won't cover creating and granting Virtual Goods, as it's detailed in another tutorial.

Log your two players into the API Explorer and register them for RTT events. Use the `FindOrCreateLobby` request, but with a slightly different set of parameters to incorporate your filter script. These parameters should include any necessary lobby settings and the DLCCode to trigger the script. This setup will ensure the filter script checks if the second player owns the required Virtual Good.

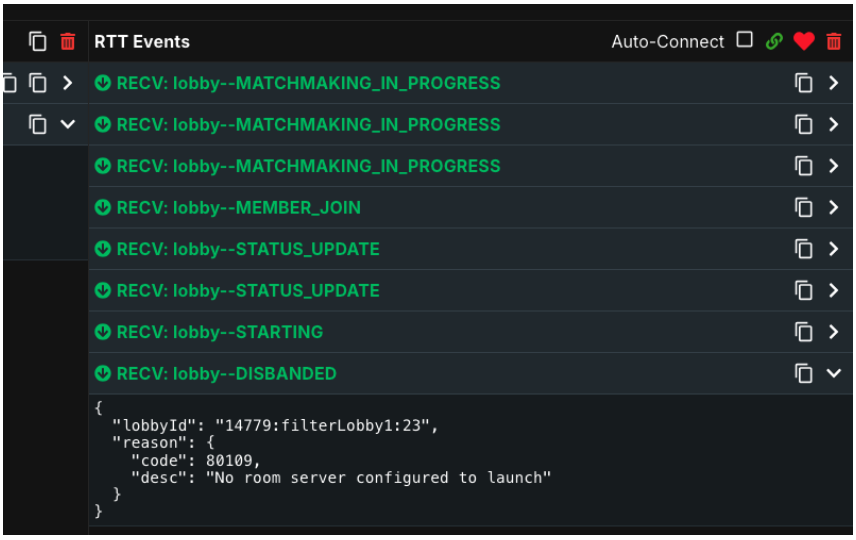
```
{
 "lobbyType": "filterLobby1",
 "rating": 100,
 "maxSteps": 3,
 "algo": {
   "strategy": "ranged-absolute",
   "alignment": "center",
   "ranges": [
     1000
   ]
 },
 "filterJson": {},
 "isReady": true,
 "extraJson": {},
 "settings": {
   "DLCCode": "DLC_1"
 }
}
```

Here we also adjust the search algorithm to have a wide range for easier match finding. Then, add a parameter called “DLCCode” to the settings JSON. This will be used by the filter script to check for the required Virtual Good during execution.

Note

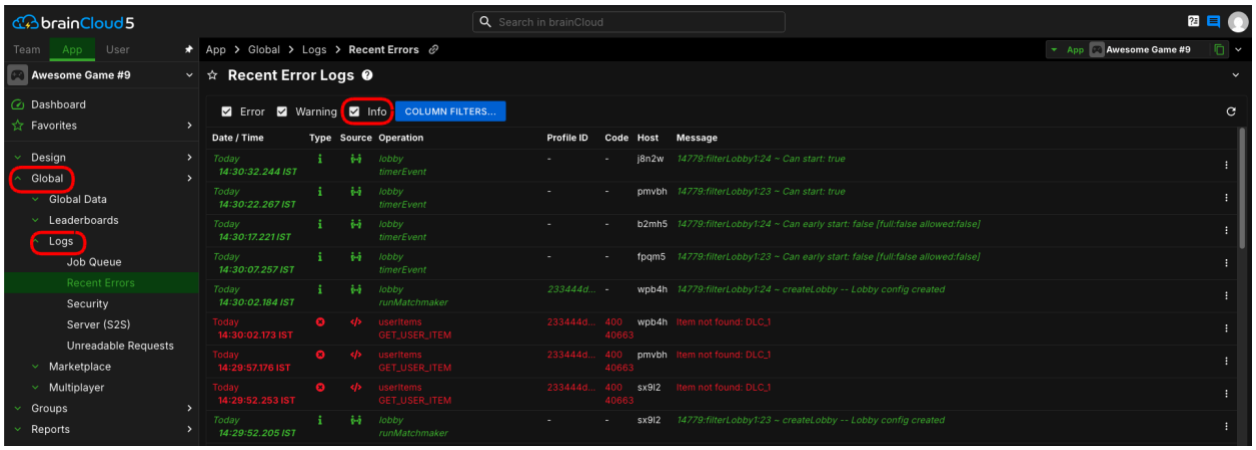
The second player doesn't need to include the DLCCode parameter in their `FindOrCreateLobby` request. They can attempt to join any lobby, while the filter script will ensure they can't access the first player’s lobby without the correct DLC.

Execute the request for both players and wait for the logs to come in and for the lobby to be disbanded.



While it may seem like the filter script worked, with both players appearing to join, the second player actually couldn't join the first player's lobby and created a new one instead.

To confirm this, check the error logs. Go to **Global → Logs → Recent Errors**. Enable Info logs to clearly see what occurred, revealing that the filter script prevented the second player from joining due to the missing DLC.



The log at 14:29:52 IST shows the first player created a new lobby. Then, three errors appear, each 5 seconds apart, indicating the "DLC_1" item was not found. After these errors, a new lobby with a different ID was created.

This sequence shows the second player attempted to join the first player's lobby at each step. The filter script ran but failed the check due to the missing item.

In order to complete this example we would need to create some Virtual Goods and grant them to the user and test again, which is already covered in Player Inventory however this example still effectively demonstrates how complex rules and behaviors can be configured in matchmaking using brainCloud filter scripts. With the flexibility of Cloud Code, you can reproduce more complex rules and behaviors needed for your migration.

PlayFab Lobbies

In this guide, we've compared PlayFab's matchmaking solutions to brainCloud lobbies and explained how brainCloud incorporates matchmaking. However, PlayFab's lobbies are separate from matchmaking, so it's important to compare these to brainCloud lobbies as well.

Essentially, both brainCloud and PlayFab lobbies allow players to gather before starting a multiplayer session. In PlayFab, players can create custom lobbies, send and accept invites, search open lobbies, update lobby properties, remove players and integrate lobbies into matchmaking flows.

brainCloud allows for most of this functionality out of the box, for example:

- CreateLobby
  Allows you to create a new lobby and provide custom settings. You can also provide a list of player's profile IDs to automatically add them to the lobby.

- GetLobbyData / UpdateSettings
  These APIs allow you to get the current state of your lobby and update any settings needed as your lobby changes.

- RemoveMember
  Allows the owner of the lobby to kick members from the lobby.

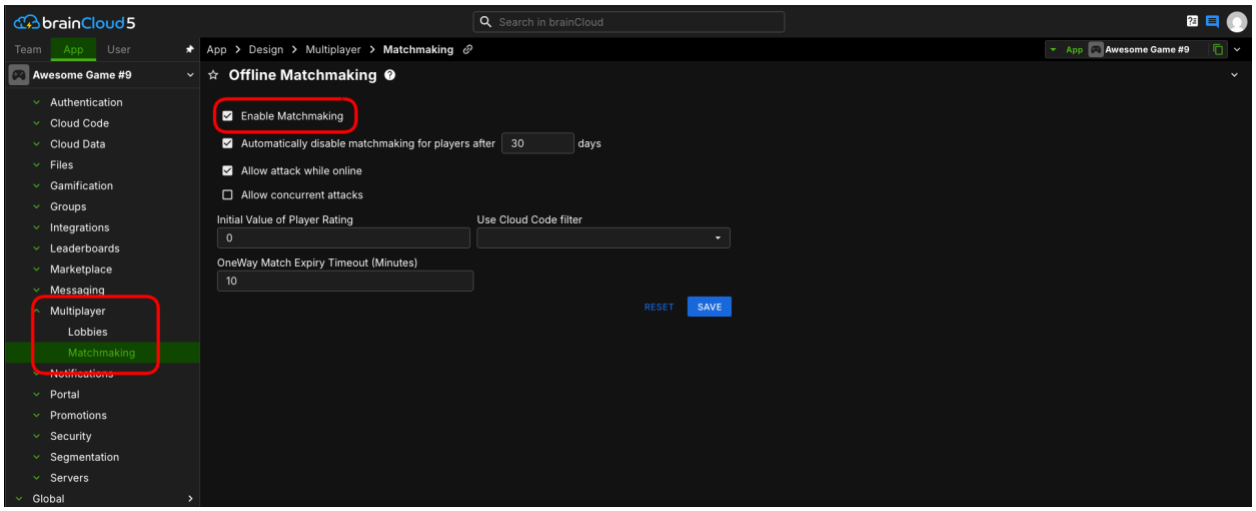
Lobby Invites

While brainCloud doesn't have a specific API for sending lobby invites, you can join any lobby with a lobby ID. To replace PlayFab's lobby invites, use the SendMessage request. Include the lobby ID and any necessary data in the message payload. The recipient can then use the JoinLobby request with the provided lobby ID.

Offline Matchmaking

brainCloud offers an offline matchmaking feature for games where players don't need to be online for multiplayer interactions. An example of this type of game might be tower defense games or Clash of Clans-like experiences.

Setting up Offline Matchmaking is simple, you first need to enable the feature from the **Design** → **Multiplayer** menu. You can read more about the settings for offline matchmaking here.



The main thing to note about offline matchmaking is that instead of submitting a ticket, or looking for a lobby, the player must instead enable matchmaking. This allows them to then be found by other players at any point. It is therefore also important to allow players the option to disable this feature if they wish.

For players that wish to find a match, there are a range of options similar to those we have already covered which allow player to find players based on a rating value, based on complex player attributes and using cloud-code or filter scripts.