

Desktop PlayFab Matchmaking Sample

*\* This sample is compatible with the June 2020 QFE 3 GDK and the Windows 10 (build 10.0.19041) SDK.*

# Description

This sample provides a simplified demonstration and example of PlayFab Matchmaking services and APIs which are accessible through the PlayFab developer portal in conjunction with the PlayFab GDK core client libraries that are available for Scarlett, XboxOne, and Desktop platforms. PlayFab provides in-depth documentation for these APIs through their documentation portal.

# Building the Sample

This sample was developed and tested against the June 2020 QFE4 release installed in conjunction with Visual Studio 2019 version 16.7.1. Whatever additional SDK components or dependencies that are required -- as mentioned during the installation of the GDK -- will have to be satisfied prior to attempting to build this sample.

Open the solution file within Visual Studio and build the solution for either the Debug or Release configuration. Make sure that the Local Machine Debugger is listed as the default debugger within the selection drop-down, and select the Desktop platform.

Update: this sample now works with Visual Studio 2017 (15.9.38) using June 2021 GDK.

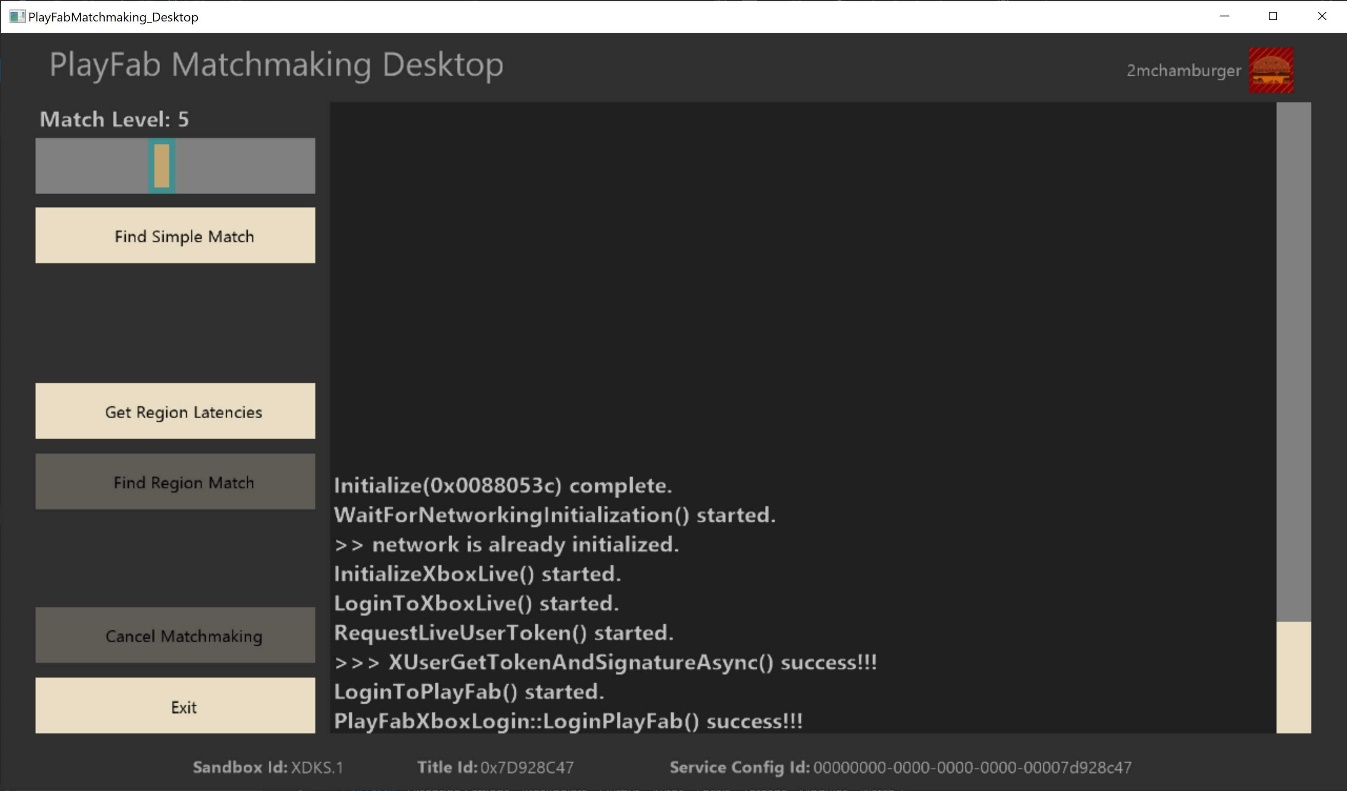
# Using the sample

The sample can be controlled either through a gamepad that is connected directly to the development desktop PC, or through the mouse & keyboard.

Upon successfully build and deployment of the sample solution, either from the Visual Studio 2019 IDE, or through the GDK toolset, a screen like that presented below should be visible. If no user is already logged in on the local sandbox XDKS.1, a standard Xbox Live login UI will present itself beforehand.

Execution details (such API results or any encountered errors) are logged into the displayed console window which supports a history of 100 log lines.

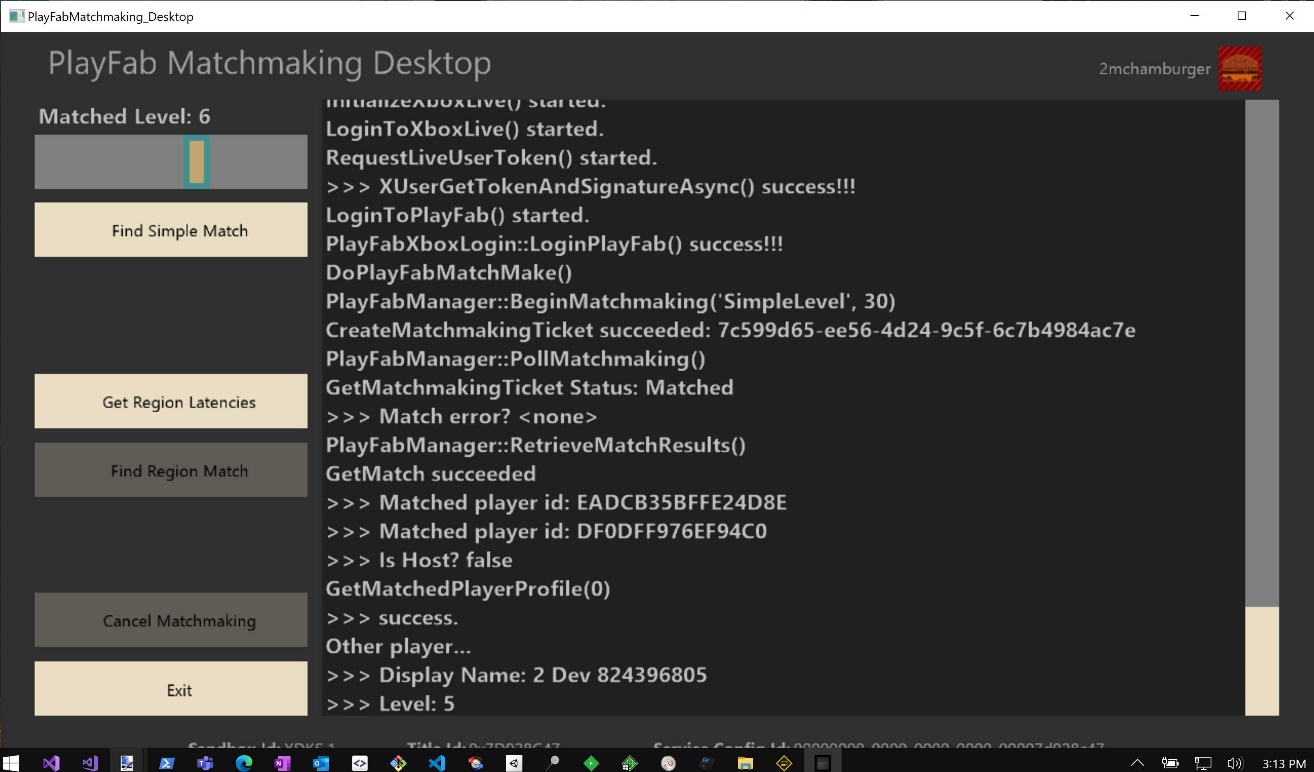
## Sample Start Screen



Before matchmaking can be executed, first the user is logged into Xbox Live and PlayFab (using Live as the identity provider) after network readiness has been established.

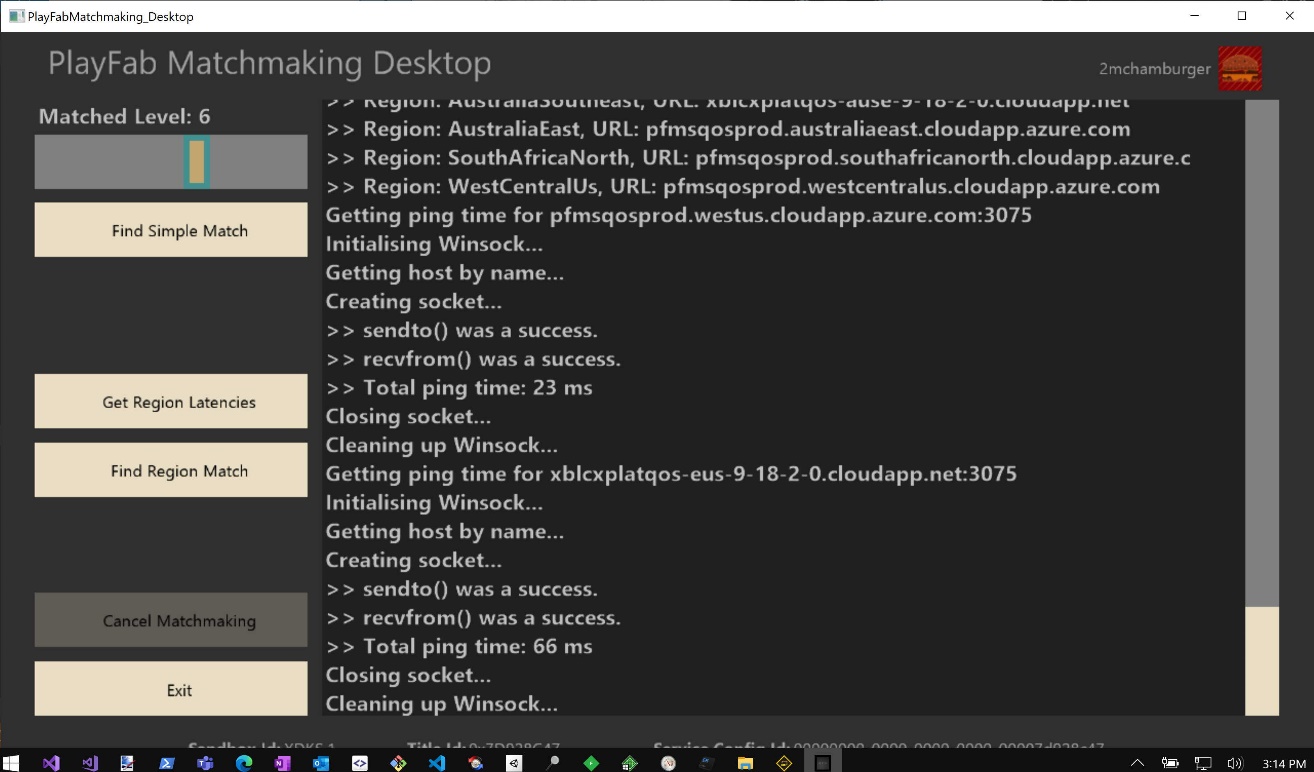
|  |  |  |
| --- | --- | --- |
| Action | Gamepad | Mouse + Keyboard |
| Change highlighted button | D-Pad | Arrow keys |
| Press highlighted button | A Button | LMB or Enter |
| Modify slider/scrollbar | Left Thumbstick | LMB + Mouse drag |
| Exit | View Button | Esc |

## Find Simple Match



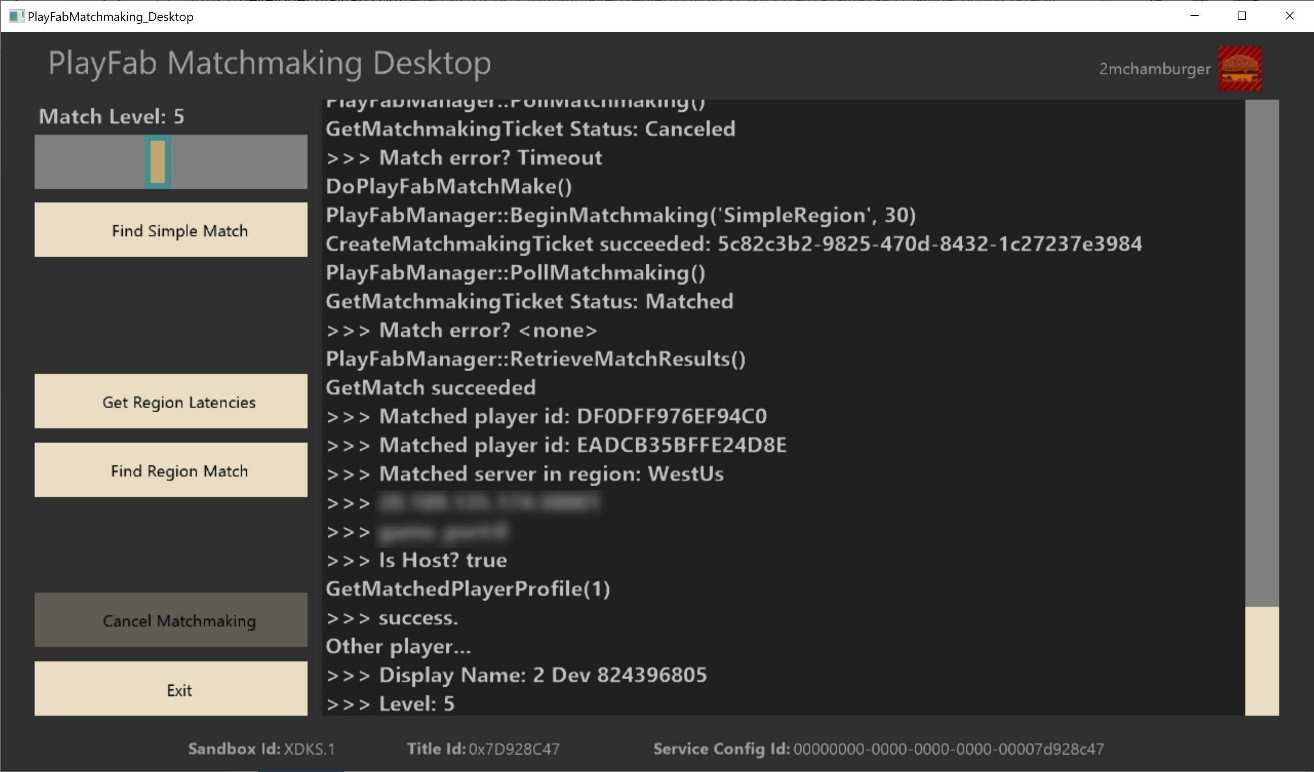
The player level slider sets the level from which to matchmaking against a queue that uses numerical difference of a User attribute to find a suitable match.

## Get Region Latencies



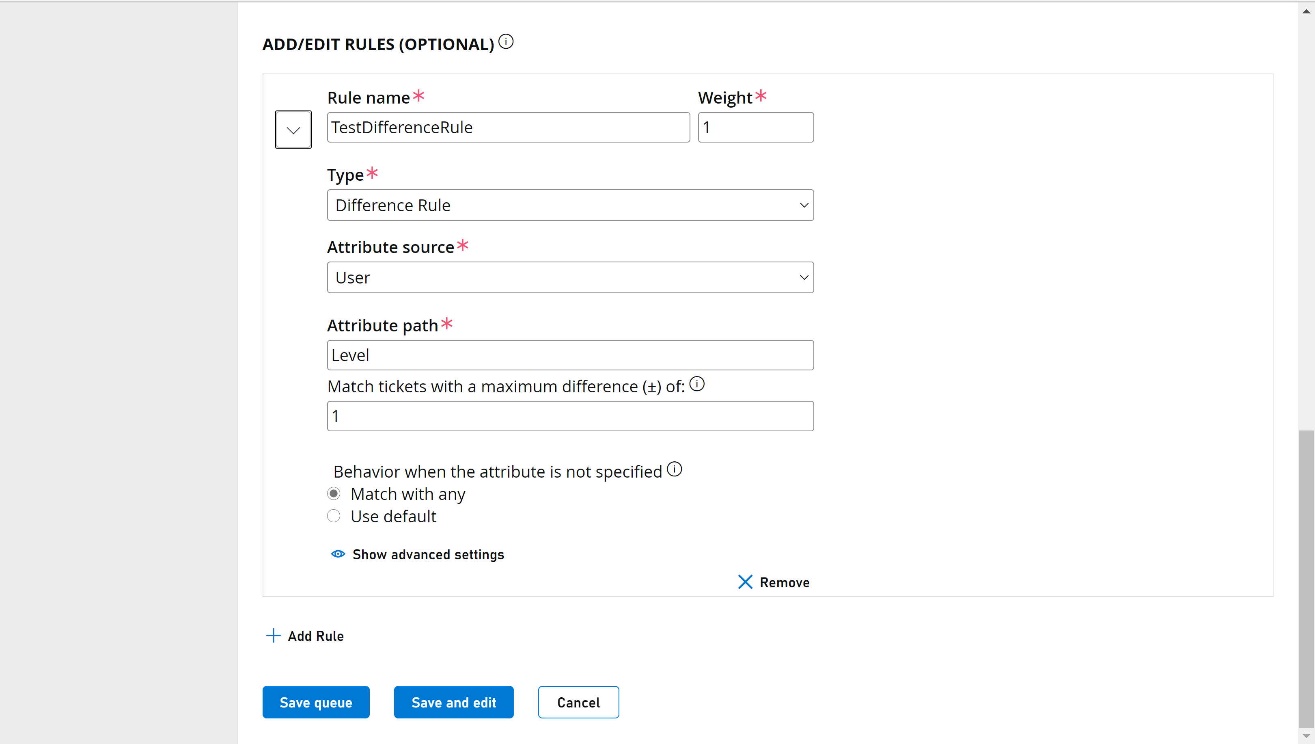
Before attempting to matchmake on a queue that employs region latency rules, first we obtain the latencies to the EastUs and WestUs datacenter beacons by timing a send & receive UDP packet.

## Find Region Match



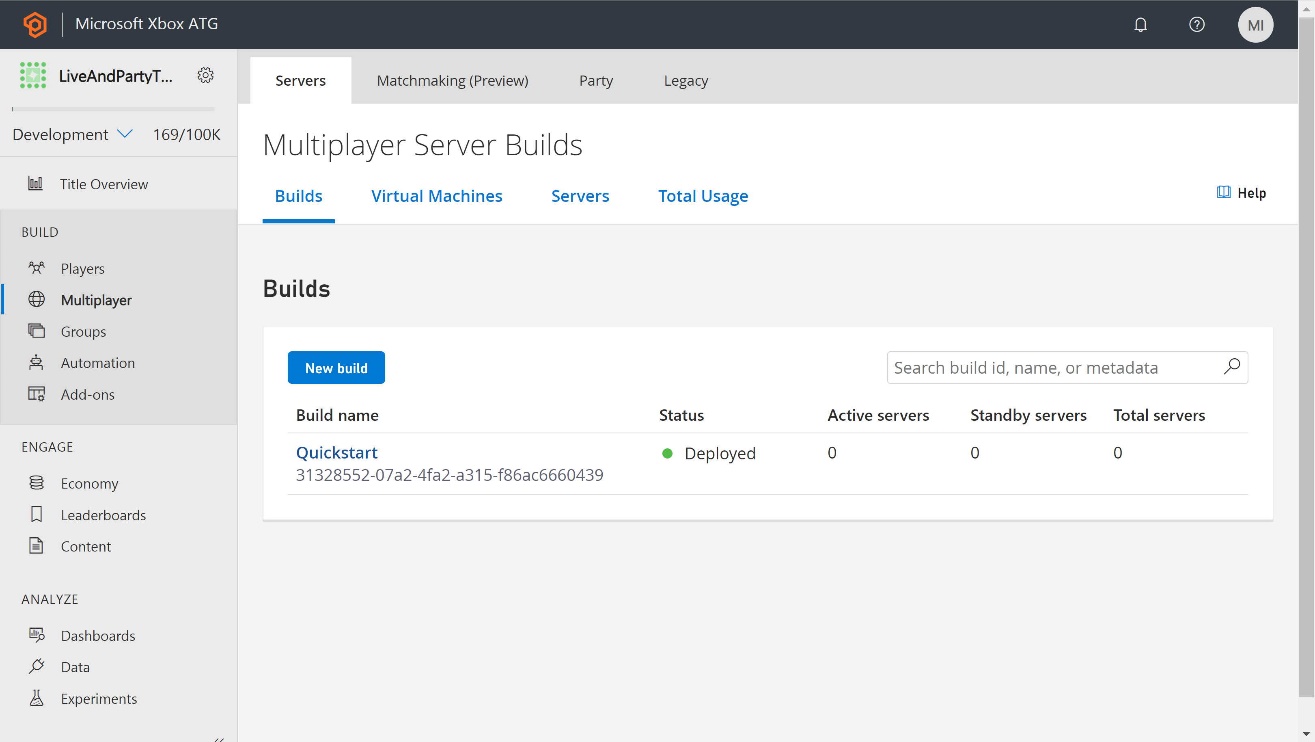
When a match is successfully created for a matchmaking ticket which targets a matchmaking queue that uses region latency rules, the server connection details are provided to the client.

## Simple Difference Rule – GameManager Matchmaking Console



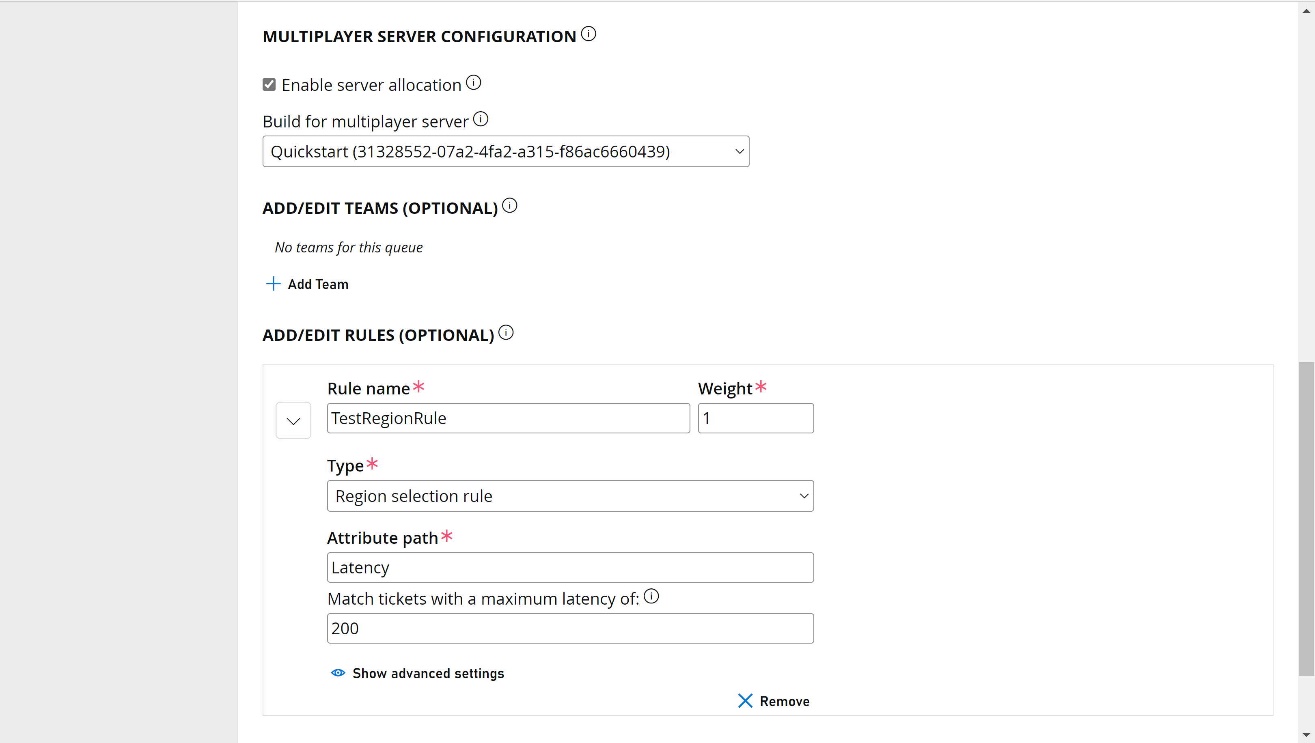
Screenshot of the PlayFab Matchmaking rule editor within the GameManager multiplayer console for setting a simple User attribute numerical difference rule for matchmaking.

## Available Server Build Pool – GameManager Server Console



Screenshot taken from the PlayFab GameManager server console for managing and deploying server pools for various datacenter regions. The PlayFab “Quickstart” server example was used.

## Server Region Latency Rule – GameManager Matchmaking Console



Screenshot showing a simplified server region matchmaking rule which relies on connection latency to find a suitable player match.

# Implementation notes

The method employed by the sample to determine the latencies to the various PlayFab region beacons is done through the use of simple UDP socket send & receive communication. Other methods suitable for Xbox certification may be employed instead, such as the use of dedicated APIs specific to this purpose.

# 

# Known issues

Attempting to perform a matchmake based on region simply through building and running the sample will fail or timeout. As demonstrated in the screenshots above, this sample does not actually deploy servers to Azure regions. To fully test this functionality, the developer must follow the guidelines set forth in the PlayFab documentation for deploying custom game servers to real Azure regions, and associating those builds with region matching rules (within queues) accordingly.

# Update history

|  |  |  |
| --- | --- | --- |
| Description | Release Date | Version |
| Initial draft of README for the sample. Includes build requirements, usage details, and notes and issues. | Sept. 1, 2020 | 1.0 |
| Updated to make compatible with VS2017 | September 2020 | 1.1 |

# Privacy Statement

When compiling and running a sample, the file name of the sample executable will be sent to Microsoft to help track sample usage. To opt-out of this data collection, you can remove the block of code in Main.cpp labeled “Sample Usage Telemetry”.

For more information about Microsoft’s privacy policies in general, see the [Microsoft Privacy Statement](https://privacy.microsoft.com/en-us/privacystatement/).