Roblox Game Server Documentation

Date: 7/28/2021

Author: Jacob Dominski

Description: The goal of this documentation to get a Roblox game server up and running without any experience. It will also be documented how to disable the profanity filter.

Step 1: Download Roblox Studio

If you haven’t installed Roblox Studio, then you are going to want to head over to <https://www.roblox.com/create> and download the Roblox Studio game engine. From there, you just go through the set up and by the end have Roblox Studio installed.

Step 2: Create the Game

Once you’ve installed Roblox Studio, open the application and it will ask you to log in with your account. From there, go the “New” category and select one of the templates. Then, Roblox Studio will generate the world for you. Now you can add anything you want to your game but we will continue with disabling the profanity filter.

Step 3: Disable the Profanity Filter

The first thing we need to do is run our game. To do that, press the “Play” button. Wait for your game to fully load and once you are inside, go into your explorer window on the right side and look for the “Chat” service. From there, open it up, and we will need to copy all the contents inside of it (ChatModules, ClientChatModules, ChatLocalization, ChatServiceRunner, BubbleChat, and ChatScript). Once copied, close your game and paste it into the “Chat” service again because the contents will automatically delete itself when you stop running the game. To disable the profanity filter, locate the ChatServiceRunner and open it. From there, we are going to open the ChatService. The we will go to line 351 where it shows the local variable runFilter set to RunService:IsServer() and not RunService:IsStudio(). All we need to do is change the value of runFilter to false.

**Before:**

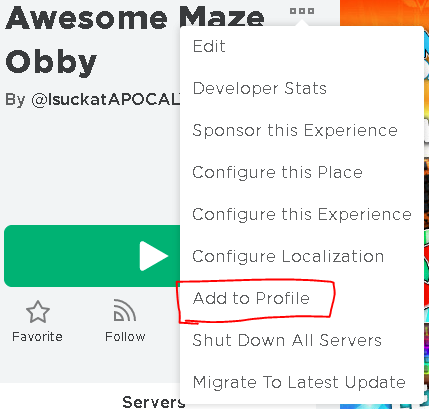


**After:**



And that’s it. You should now be able to type anything you want in your Roblox game. The only step left is to publish it.

Step 4: Publish Your Game

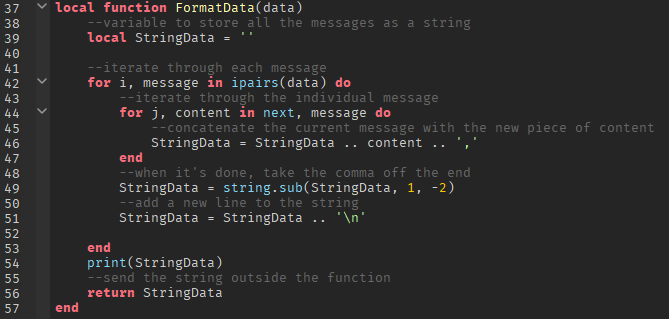
What publishing means is that your game will be hosted on Roblox servers so other people can play your game. To accomplish this, got to File > Publish To Roblox As. Once opened, a screen will appear and we will select the “Create new game” option at the bottom left of the window. From there, you can add a title to your game and a description. Once you click the create button, Roblox should publish your game. At this point your game is set to private and to view it on the website, first restart Roblox Studio and open the game again. Then, at the top right next to your username, there is a button with three dots and two lines connecting them. Click on it and it should ask you to make your game public. Click on game settings and go to the permissions tab. From there, you can set the playability to public. Once changed, click save at the bottom right. Lastly, we can go back to the three dots and two lines button next to your name and select the copy link. We can paste that link into any web browser and our game is now public. By default your published game is not attached to your profile so to do that, click the three dots adjacent to the title for your game and select “add to profile”. 

Step 5: Store User Messages

In this step, we will be storing the user messages on the server until all the players leave. The easiest and most memory efficient way would be to store the messages in a data structure. First, we need to create a script in the ServerScriptService (just a regular script). In there, we can create an array to store all the messages and for each message, it will be a dictionary storing a unique id, player name, time, and message. We will also have a function that inserts a dictionary into the array every time a player sends a message. We can see how that would look here. We will also need to get the HttpService and the Players Service.



Additionally, when we’re ready to send the data, we need to convert it to a string following a comma separated value format (csv). To do that, we can create a function that parses the data and adds a comma after every value and a new line for every message.



Step 6: Create PUT Request

Now that the data can be properly formatted, we need a function to handle PUT requests to the S3 Bucket.



Lastly, we need a function that will call the PUT when the server shuts down.

