Twitch Chat Connect - Documentation v1.3.0

Table of contents

Main feature

Twitch Chat Connect is a client that can be used to connect to any Twitch chat using IRC protocol. It has an easy usage by using callbacks to know whenever a chat message is sent.

Setup

OAuth Token

The first thing to do is to get an OAuth token to be able to connect to the Twitch Chat. In order to do that, go to https://twitchapps.com/tmi/ and log-in with your username/password of Twitch and then you will get your token. It's recommended to use a secondary account.

Configuration

It is necessary to create an instance of `TwitchConnectConfig` with some data in order to connect to the Twitch chat.
There is a class called `TwitchConnectData` which is a Scriptable Object therefore it's possible to create an asset with the configuration.

Necessary data:
Username: Username which was used to generate the OAuth token.
UserToken: OAuth token generated in the previous step.
ChannelName: Name of the channel you want to connect to.

Usage

Adding the client

Add an empty object in your scene and add the component `TwitchChatClient`. The component is a singleton and also it's not destroyed when loading a new scene. It's only necessary to add it once in your whole game. It could be added in multiple scenes without any problem. Optionally add a reference to a `TwitchConnectData` asset with the connection data, otherwise it is necessary to create an instance of `TwitchConnectConfig`.

The component has the following fields that can be modified in the *Inspector*.

Name	Description	Default value
`Init Twitch Connect Data`	(optional) Reference to an asset of type TwitchConnectData with the necessary data to connect to the Twitch Chat.	*<empty>*
`Command Prefix`	All messages starting with this prefix will be recognized as commands. All other messages will be ignored.	**`!`**

Available classes & methods

`TwitchChatClient.instance.Init(OnSuccess onSuccess, OnError onError)`
This method is necessary to connect to Twitch's chat. It receives two parameters which are callbacks *onSuccess()* and *onError(string)*.
The field *Init Twitch Connect Data* needs to have a valid reference.

`TwitchChatClient.instance.Init(TwitchConnectConfig twitchConnectConfig, OnSuccess onSuccess, OnError onError)`
Same as the previous method, the difference is that the configuration is given in the first parameter ***`twitchConnectConfig`***.

`TwitchUser` ***`TwitchUserManager.GetUser(string username)`***
Receives an username and returns a `TwitchUser` with user information.

`bool` ***`TwitchUserManager.HasUser(string username)`***
Receives an username and returns *true/false* if the user is in the chat.

`List<TwitchUser>` ***`TwitchUserManager.Users`***
Returns a list of all connected users to the chat.

Receiving messages

In order to receive messages, It's necessary to subscribe to different events.
`TwitchChatClient.instance.onChatCommandReceived(`*`TwitchChatCommand`****`)`***
This is an event that will be triggered when a chat user sends a message started by the *command prefix* defined in the component **`TwitchChatClient`**.

`TwitchChatClient.instance.onChatRewardReceived(`*`TwitchChatReward`****`)`***
This is an event that will be triggered when a chat user unlocks a reward.

`TwitchChatClient.instance.onChatMessageReceived(`*`TwitchChatMessage`****`)`***
This is an event that will be triggered when a chat user sends a normal message.

TwitchChatMessage is a POCO class that contains twitch chat message information.

Name	Description	Type
User	User who sent the message.	TwitchUser
Message	Message sent	string
Bits	Amount of bits sent in the message.	int

TwitchChatCommand extends ***TwitchChatMessage*** and processes the message to provide information about the *command* and the *parameters* used.

Name	Description	Type
command	Command sent with the prefix included.	string
parameters	Array of strings as result of splitting the message with a space, excluding the command.	string[]

TwitchChatReward extends ***TwitchChatMessage*** and adds the custom reward id.

Name	Description	Type
CustomRewardId	Custom reward id	string

TwitchUser is a POCO class that contains the twitch user information. The **Username** is always available but the rest of the data is only available if the user sent at least one message.

Public attributes/methods

Name	Description	Type
Username	Twitch username.	string
Id	Twitch user ID	string
DisplayName	Twitch displayname (returns **Username** as default)	string
Badges	List of user badges.	List<TwitchUserBadge>
IsSub	Returns *true* when the user is subscribed.	bool

IsTurbo	Returns *true* when the user is turbo.	`bool`
IsVip	Returns *true* when the user is vip.	`bool`
IsModerator	Returns *true* when the user is moderator.	`bool`
IsBitsLeader	Returns *true* when the user is a bit leader.	`bool`
IsBroadcaster	Returns *true* when the user is the broadcaster.	`bool`
IsBits	Returns *true* when the user gave bits at some point.	`bool`
GetBadge()	Returns a **TwitchUserBadge** given a valid badge name.	`TwitchUserBadge`
HasBadge()	Returns *true* when the user has the given valid badge name.	

TwitchUserBadge is a POCO class that contains user's badge information.

Name	Description	Type
Name	Badge name. Examples: *admin*, *bits*, *broadcaster*, *moderator*, *subscriber*, *staff*, *turbo*, *vip*	`string`
Version	Version name. Check official documentation for more information.	`int`

Example

For a full working example in Unity, open the scene **SampleScene** in the directory *Example*.

```
public class Example : MonoBehaviour
{
    void Start()
    {
        TwitchChatClient.instance.Init(() =>
        {
                TwitchChatClient.instance.onChatMessageReceived += ShowMessage;
                TwitchChatClient.instance.onChatCommandReceived += ShowCommand;
                TwitchChatClient.instance.onChatRewardReceived += ShowReward;

        }, message =>
        {
            Debug.LogError(message);
        });
    }

    void ShowCommand(TwitchChatCommand chatCommand)
    {
    }

    void ShowReward(TwitchChatReward chatReward)
    {
    }

    void ShowMessage(TwitchChatMessage chatMessage)
    {
    }

}
```