May 29, 2026

How to make Discord widgets

A while ago Discord added a feature called "Widgets" which allowed certain games (namely Wuthering Waves and Marvel Rivals) to display stats on your profile. Now fairly recently they added an experiment called "widgets v2" which allows any Discord application (bot) to create these widgets. The method to create these have been fairly private, mostly because it's annoying, but now that someone has made a PAID service to create these I felt like there needed to be an actual guide to prevent this extremely scummy behavior from getting traction.

Special thanks to @aamia, @bignutty and @.dziurwa and the rest of the smart people at Discord Previews for getting out the first info on these.

Note that this is not "the best" way of creating these, just *a* way to create these. **You will also need programming knowledge and more in-depth knowledge with your browsers' developer tools. I will not act as personal support for this.**

Setting up your Application and Developer Portal

the new application head over to Games then Social SDK on the sidebar. You will now need to fill out the form to get access to the Social SDK, which widgets v2 is part of. You do not need an actual game or anything for this, and you will get access instantly after pressing submit.

After this you will want to open your browsers' developer tools. We need to give ourselves access to an experiment which is only available on the Developer Portal to access the widget config editor. Open the console tab and run the following code:

```
let _mods = webpackChunkdiscord_developers.push([[Symbol()],{},r=>r.c]);
webpackChunkdiscord_developers.pop();

let findByProps = (...props) => {
    for (let m of Object.values(_mods)) {
        try {
            if (!m.exports || m.exports === window) continue;
            if (props.every((x) => m.exports?.[x])) return m.exports;

            for (let ex in m.exports) {
                if (props.every((x) => m.exports?.[ex]?.[x]) && m.exports[ex][Symbol.toStringTag] !== 'IntlMessagesProxy') return m.exports[ex];
            }
        } catch {}
    }
}

findByProps("getAll").getAll().find(e=>e.getName() === "ApexExperimentStore").createOverride("2026-03-widget-config-editor", 1)
```

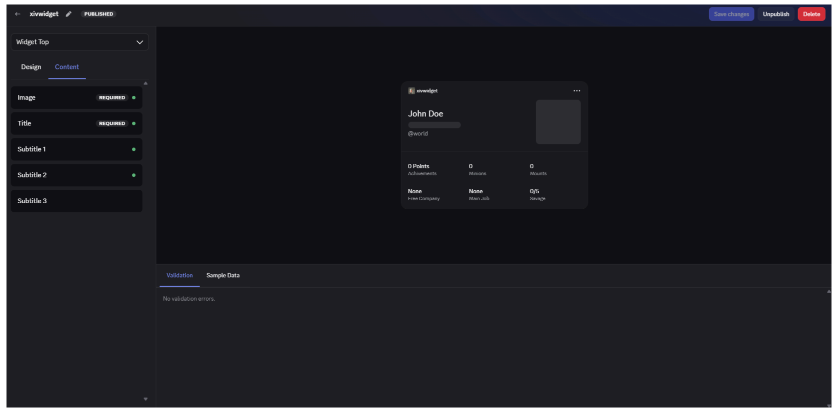
After running the code click the back arrow at the top left and then open the page for your application again (do not refresh the page as this will remove the override again), you will now see the Widget page under the Games section in the sidebar. Here you can now create your widget.

Creating your Widget

After clicking on Create widget you will be brought to the widget editor. This is a pretty easy to use editor so you shouldn't have much trouble here, but there are a couple of things to explain.

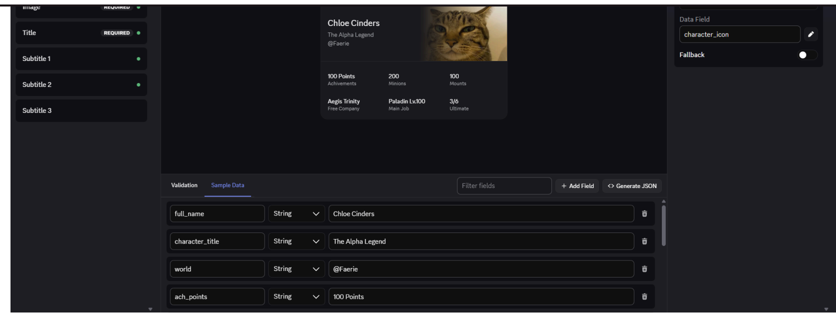
required fields.

Fields can have one of 2 types: Custom/Asset and User Data (though Number is always locked to User Data). Custom String/Application Asset are static and never change. If you set your Widget Top image to an Application Asset everyone will have that exact asset. If you change the title to a Custom String everyone will have that exact string there. User Data however lets you change fields based on the user. If I want the users' username of my service to be displayed as the Title on the Widget Top of the widget, I set the field to "Text", the Value Type to "User Data" and then the Data Field to "username" (or anything really, you can change it depending on what you want to put here). These should also have a fallback which is displayed when your application has not sent any information about the user yet. You should now style your widget how you want it. As an example I have made a widget for the game Final Fantasy XIV Online:



Now to fill it with the users' information!

Click on the "Sample Data" tab at the bottom, or click on the pencil icon next to a User Data field. Here you can provide the Widget with the data to fill out your widget, to see how a widget fully filled out with information would look. You should add every field here and some demo data:

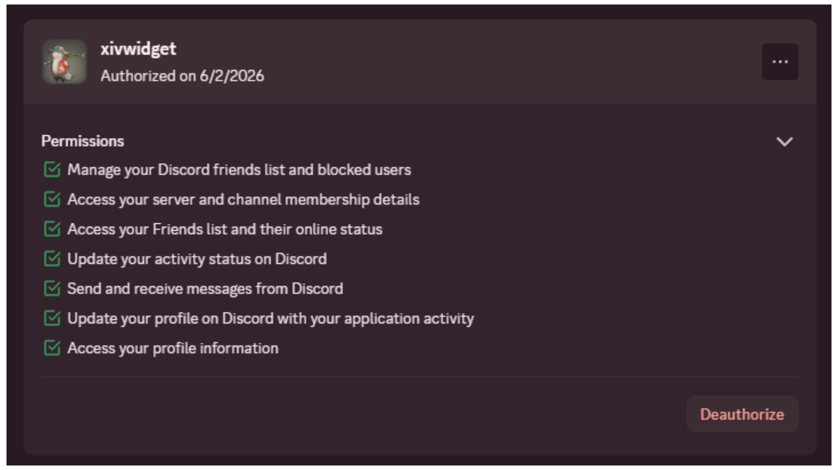


If everything looks right you can now click on "Generate Json" at the top right of the Sample Data window. Save the JSON somewhere, close the modal and then click "Save Changes" and "Publish" at the top right of your screen. Head over to your Applications page on the Developer Portal and go to the OAuth2 page under Overview in the sidebar. You will have to create a redirect URI, either point it to your own website, or what I did, point it to Discord's homepage: `https://discord.com`. Click Save at the bottom of your screen and then scroll down to the OAuth2 URL Generator. We are doing this because users need to give us permission to change things on their widget. Check the Scopes "openid" and "sdk.social_layer". Select your URI as the redirect URI and copy the generated URL. Save it somewhere with the JSON we just saved from the Sample Data. Change the `response_type=code` to `response_type=token` inside the URL then open it once to check if there is no error, if you get invalid scopes make sure you have the Slayer SDK form filled out from step 1 of the guide.

Application Identities

If you are looking to make a service for other people please skip this chapter, **if you are looking to create a widget for yourself only then this chapter is REQUIRED**. Ignoring this step causes the widget to display "Your game stats are still syncing. Keep playing!" message and not display to other users.

Check if you have successfully authenticated with your application from the end of the previous chapter. Go to the "Authorized Apps" page in the Discord settings and search for your application. It must have the permissions: "Send you direct messages", "Manage your Friends list and their online status", "Update your activity status on Discord", "Send and receive messages from Discord" and "Access your profile information".



We now have to issue an identity. Go back to your application in the developer portal, go to the Bot page and reset the Token. Copy the new token into the place where you saved the sample data from earlier. Edit the sample data to your liking and add a "username" field to it at the root with any value, open the console in your browsers' developer tools and run:

```
JSON.stringify(sample data here)
```

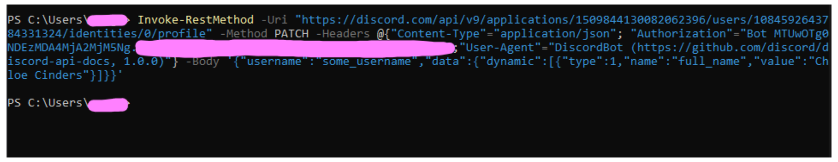
Example:

```
JSON.stringify({
    "username": "some_username",
    "data": {
        "dynamic": [
            {
                "type": 1,
                "name": "full_name",
                "value": "Chloe Cinders"
            },
        ]
    }
});
```

This will give you a JSON string, copy it and save it with your bot token. Now to send the request to issue the identity. Open Windows PowerShell on your computer (Linux users know the Linux alternative here) and run this command, replacing discordApplicationId with your applications ID, discordUserId with your own Discord user ID, botToken with the bot token you copied earlier and jsonString with the JSON string you copied earlier:

```
tities/0/profile -Method PATCH -Headers @{"Content-Typ
e"="application/json"; "Authorization"="Bot {botToke
n}";"User-Agent"="DiscordBot (https://github.com/discor
d/discord-api-docs, 1.0.0)"} -Body '{jsonString}'
```

Example:



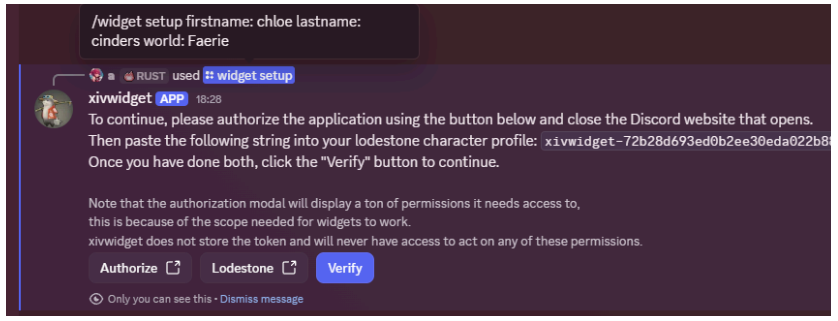
If you get no errors then you can skip the Widget Management using Bots chapter and move on straight to the Displaying the Widget on your Profile chapter!

Widget Management using Bots

Now we will have to create a Discord bot that handles user interactions to link/refresh widgets. What I and a couple other people, such as @bignutty, did was to make a User Installation Discord bot. This bot has two commands: /widget setup and /widget refresh. I made this bot in C# due to Final Fantasy XIV Online libraries usually using C#. The /widget setup command would instruct the user to authorize the application and then for my case verify their Final Fantasy XIV Online character:

```
[SlashCommand("setup", "Setup the widget")]
public async Task SetupAsync(
    [Summary(description: "First name of the characte
r")] string firstname,
    [Summary(description: "Last name of the character")]
string lastname,
    [Summary(description: "World of the character"), Aut
ocomplete(typeof(WorldAutocompleteHandler))] string worl
d)
{
    ...
    var authorizeButton = new ButtonBuilder()
    {
        Style = ButtonStyle.Link,
        Label = "Authorize",
        Url = $"https://discord.com/oauth2/authorize?cli
ent_id=1509844130082062396&response_type=token&scope=ope
nid+sdk.social_layer"
    };

    var lodestoneButton = new ButtonBuilder()
    {
```

```
e/character/{lodestoneId}/"
    };

    var verifyButton = new ButtonBuilder()
    {
        Style = ButtonStyle.Primary,
        Label = "Verify",
        CustomId = $"verify:{lodestoneId}"
    };

    var row = new ActionRowBuilder().AddComponents(autho
rizeButton, lodestoneButton, verifyButton);
    var components = new ComponentBuilder().AddRow(row).
Build();

    ...

    await RespondAsync($"""
    To continue, please authorize the application using
the button below and close the Discord website that open
s.
    Then paste the following string into your lodestone
character profile: `{verificationString}`.
    Once you have done both, click the "Verify" button t
o continue.

    -# Note that the authorization modal will display a
ton of permissions it needs access to,
    -# this is because of the scope needed for widgets t
o work.
    -# xivwidget does not store the token and will never
have access to act on any of these permissions.
    """, ephemeral: true, components: components);
}
```



The second command /widget refresh is a bit more complicated. This is what handles sending the data over to Discord for the widget to display correctly (for my bot this is also run right after verification once). Basically you have to send the JSON sample data we copied earlier with the correct information to `PATCH https://discord.com/api/v9/applications/{discordApplicationId}/users/{discordUserId}/identities/{identityId}/profile`. In this URL you replace `discordApplicationId` with the ID of your Application, `discordUserId` with the ID of the user who ran the command and then `identityId` with a unique ID. The identityId is kind of a hassle though, I recommend just always setting it to `0` which works completely fine. In my case the full URL would be:

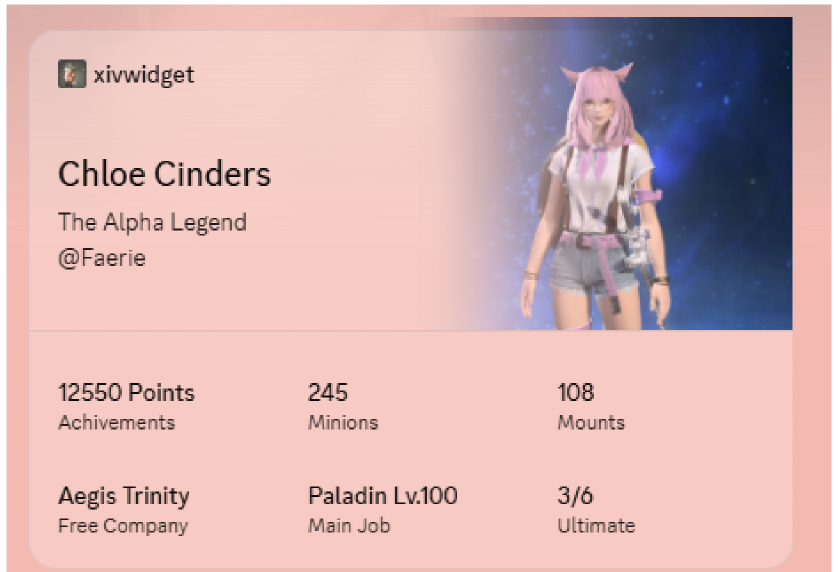
4331324/identities/0/profile. If the user has authorized the Application with the slayer scope then this request will succeed. Sample code from my bot:

```
private async Task SyncUserDiscordWidget(ulong discord, string lodestoneId, LodestoneCharacter character)
{
    ...

    var dynamicData = new List<object>
    {
        new { type = 1, name = "full_name", value = character.Name },
        new { type = 1, name = "character_title", value = character.Title },
        new { type = 1, name = "world", value = $"@{character.Server}" },
        new { type = 1, name = "ach_points", value = $"{achievementPoints} Points" },
        new { type = 2, name = "minions", value = minionCount },
        new { type = 2, name = "mounts", value = mountCount },
        new { type = 1, name = "fc", value = character.FreeCompany?.Name ?? "None" },
        new { type = 1, name = "job", value = !string.IsNullOrEmpty(job) ? $"{job} Lv.{jobLevel}" : "Not set" },
        new { type = 1, name = "raids", value = $"{raidCleared}/{raidTotal}" },
        new { type = 1, name = "raid_label", value = raidLabel }
    };

    if (character.Portrait != null)
    {
        dynamicData.Add(new { type = 3, name = "character_icon", value = new { url = character.Portrait } });
    }

    var payload = new
    {
        username = character.Name,
        data = new
        {
            dynamic = dynamicData
        }
    };

    var json = JsonSerializer.Serialize(payload);
    var content = new StringContent(json, System.Text.Encoding.UTF8, "application/json");

    var clientId = "1509844130082062396";
    var url = $"https://discord.com/api/v9/applications/{clientId}/users/{discordId}/identities/0/profile";

    using var httpClient = new HttpClient();
    httpClient.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bot", token);

    var response = await httpClient.PatchAsync(url, content);
    if (!response.IsSuccessStatusCode)
    {
        var resText = await response.Content.ReadAsStringAsync();
        throw new Exception($"Discord API error: {resText}");
    }
}
```

Displaying the Widget on your Profile

Discord made some changes to how you add widgets to your profile. It is quite annoying now. Basically you would need to add the widget to your profile via the API. To make this easier though please join the Discord Previews Discord server and head over to this thread: https://discord.com/channels/603970300668805120/1509942620762276011. Here you will find snippets which you run in the Developer tools of your client/browser to either directly add the widget to your profile or add it to the "Add Widget" menu at the top right of your Discord profile. Make sure to also have the 2026-03-application-widget-v2-renderer set to Variant 1. If you don't know how to do this please ask in the Discord Previews general channel.

If everything worked you can now see your widget on your profile!



Now you know how to make your own widgets!

Please use this responsibly, abuse will likely get the feature taken away from *everyone*. Meaning don't be an idiot and make widgets that make you out as Discord staff or spam stuff like meow or whatever. Making a examples there's:

- Gramophone by @bignutty
- AniList by @.dziurwa
- osu! by @.dziurwa
- Steam profile by @wooslow
- xivwidget by me!

The code for xivwidget can also be found here if you need any examples coding the bot: https://github.com/chloecinders/xivwidget

Happy Developing!