Player Prefs X

This is an extended version of Unity's `PlayerPrefs`, designed to handle more than just `int`, `float`, and `string`. It supports custom classes, arrays, lists, and more complex data types while still using Unity's `PlayerPrefs` internally.

It works by serializing non-primitive data into JSON format using Unity's `JsonUtility`. If Unity's serializer can handle it, `PlayerPrefsX` can store it. You also get a simple API with just one generic method for setting values and one for retrieving them (with an optional default fallback).

See the repository of this project on:

- https://github.com/demircialiihsan/unity-player-prefs-x

Installation

You can:

- Clone or download this repository and copy/move the `PlayerPrefsX` folder into your project.

Or:

- Download the *.unitypackage* file from the releases section and import it into your project.

After importing, you can move the `PlayerPrefsX` folder anywhere within the `Assets` folder. It doesn't need to be at the root of the `Assets` directory.

Supported Data Types

- `int`, `float`, `string` (standard `PlayerPrefs` types)
- `bool`
- Plain `class` and `struct` with the `[Serializable]` attribute that contain fields supported by the Unity serializer.
- ***Arrays*** and ***Lists*** of any supported type

How To Use

API

Include `UnityPlayerPrefsX` namespace to access `PlayerPrefsX` :

- `using UnityPlayerPrefsX;`

`PlayerPrefsX`

`void Set<T>(string key, T value)` : Sets the value for the preference identified by the given `key` .

`T Get<T>(string key)` : Returns the value corresponding to `key` in the preference file if it exists. Otherwise, returns `default` of type `T` .

`T Get<T>(string key, T defaultValue)` : Same as *T Get(string key)*, but if the key doesn't exist, returns the given `defaultValue` as is.

The following are the redirection methods. They just call the corresponding functions with the same names of standard `PlayerPrefs` .

`void DeleteAll()`

`void DeleteKey(string key)`

`bool HasKey(string key)`

`void Save()`

> Since this system uses `PlayerPrefs` internally, do not use the same keys you have used in `PlayerPrefs` .

Example code

```
void Example()
{
    // if the key doesn't exist;
```

```
    // returns the default of int, 0
    var number = PlayerPrefsX.Get<int>("number");

    // returns specified red color
    var color = PlayerPrefsX.Get("color", Color.red);

    // returns null
    Vector3[] points = PlayerPrefsX.Get<Vector3[]>("points");

    points ??= new Vector3[]
    {
        new(0, 1, 0),
        //...
    };

    PlayerPrefsX.Set("number", number);
    PlayerPrefsX.Set("color", color);
    PlayerPrefsX.Set("points", points);
}
```

You can also explore the example usage of saving a list of custom data classes in the `PlayerPrefsX/Samples` folder. You can safely delete this folder once you no longer need it.

Editor

You can access *PlayerPrefsX* menu from **'Tools > PlayerPrefsX'**.

Clearing the Data

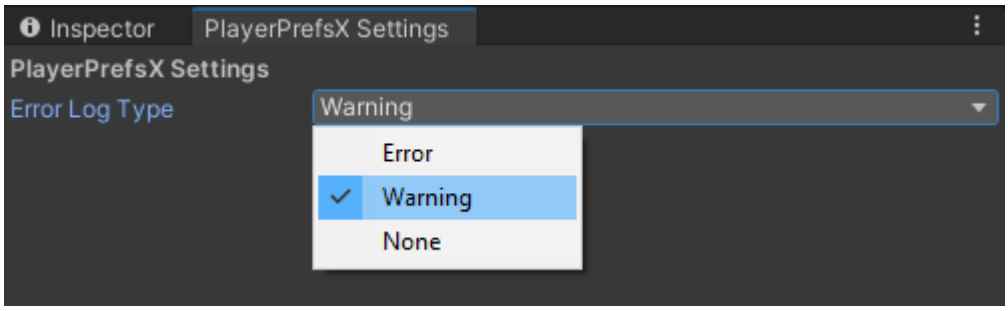
Use **'Tools > PlayerPrefsX > Clear All PlayerPrefsX'** to clear all the data.

> Clearing all *PlayerPrefsX* data will clear all *PlayerPrefs* data.

Error Handling

If a saved JSON entry becomes corrupted, it won't load properly and may throw errors. You can set *Error Log* Type to *Warning* or *None* to avoid breaking your game:

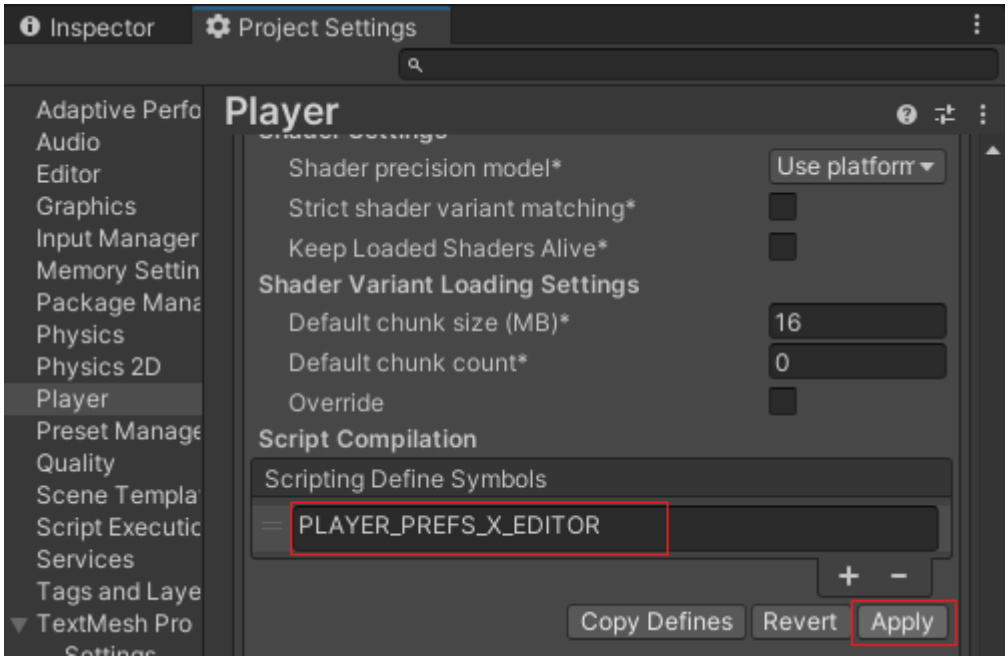
1. Open settings from **'Tools > PlayerPrefsX > Open Settings'**
2. Select the error log type from the dropdown.



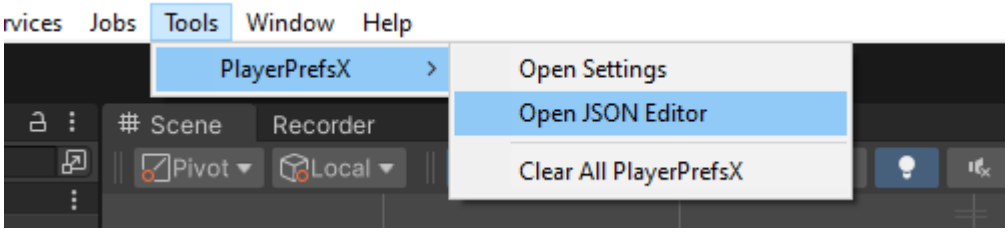
Advanced: JSON Entry Editor

You can enable JSON Editor to edit JSON-formatted PlayerPrefs data. This requires Newtonsoft-JSON package to be installed.

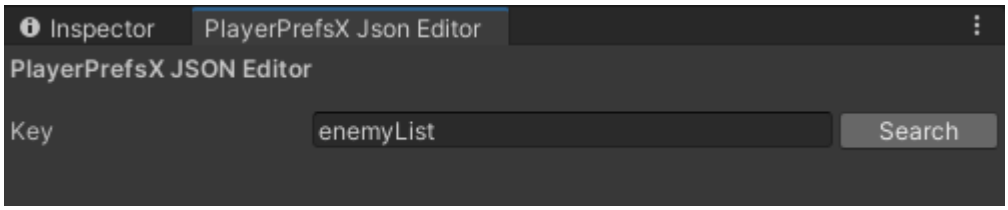
1. Install the package `com.unity.nuget.newtonsoft-json` (via Unity's Package Manager).
2. Add `PLAYER_PREFS_X_EDITOR` to **'Project Settings > Player > Scripting Define Symbols'**.
   Make sure to click *Apply*.



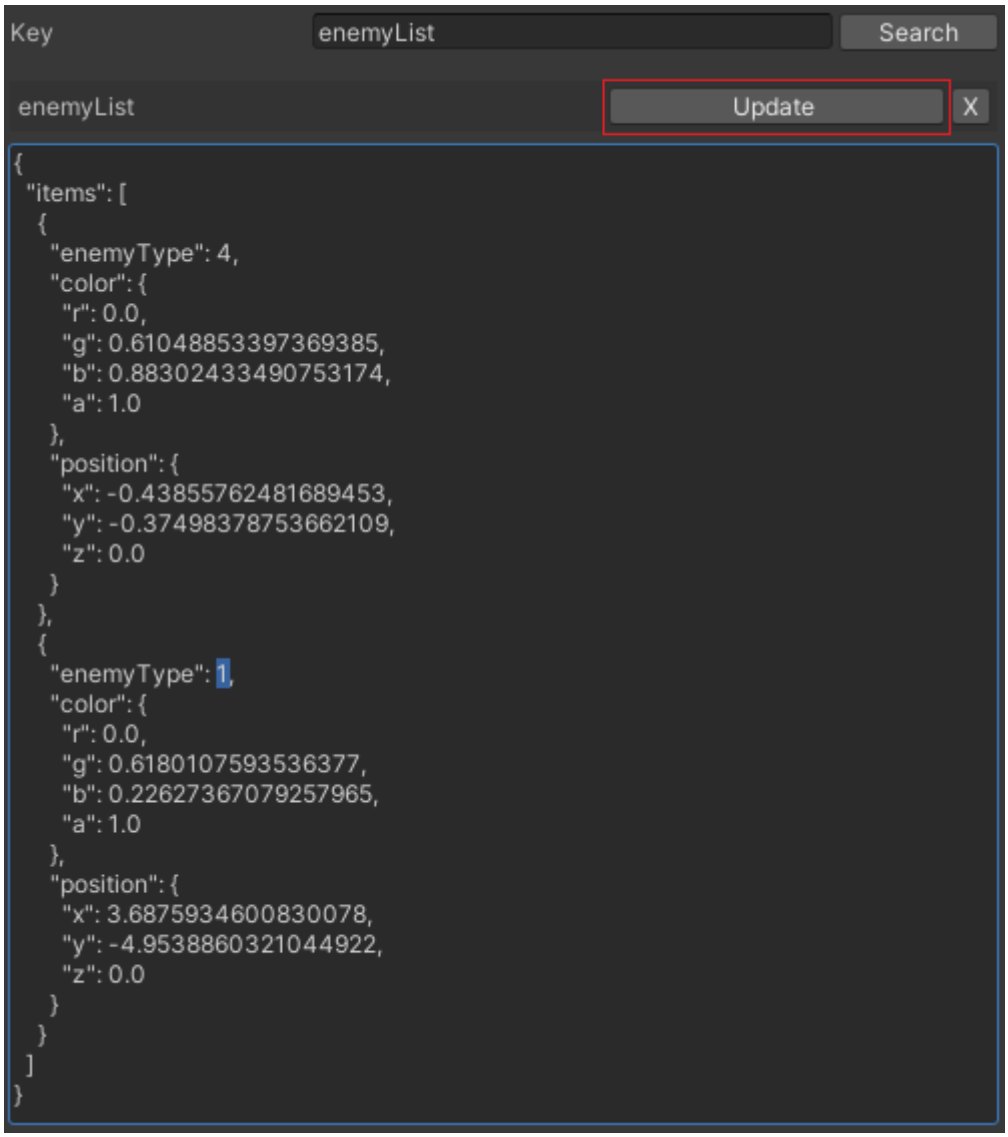
3. A new menu option **'Tools > PlayerPrefsX > Open JSON Editor'** will appear. Open the JSON editor.



4. Enter the ***key*** for your preference and click the *Search* button.



5. Edit the JSON displayed in the text area and click *Update*.



> The *JSON Editor* won't let you update a preference with invalid JSON. Still, it is meant to be used for testing purposes. Use it with caution.